TECHNIQUES IN IMAGE RESTORATION AND ENHANCEMENT

John Stuart Andrew Hepburn

Submitted for M.Sc.(Eng.) Degree

University of Cape Town

April 1975

CONTENTS

FIGURES

PLATES

INTRODUCTION CHAPTER 1

Image processing is a rapidly expanding field with contributions from a variety of disciplines such as electrical engineering, physics and medicine. As a result there has been little coordination in its progress, each group applying methods with which it is most familiar, though the systems approach of electrical engineering is gaining in popularity. The advantage of tackling a field from many angles is that the best approach is more likely to be found in the end, but at the same time there is often a duplication of effort by the pursuit of methods which appear different superficially, but in effect are equivalent, as happened in Quantum Mechanics with the Schrödinger and Heisenberg analyses.

Image processing in its broad sense pervades many areas but it is convenient to group it into three main sections, *viz:* image coding, usually for image transmission over telecommunication links; pattern recognition for detecting the presence of a particular distribution in an image which is generally corrupted to some extent by noise; and image restoration, which aims to recover a faithful reproduction of a perfect image which has been degraded, and image enhancement which attempts to present an image in a form which will convey most readily the desired information to the human brain and takes account of the characteristics of vision.

It is the object of this thesis to investigate some of the techniques of image restoration and enhancement.

There are many different media for implementing the various processes, but digital computation and coherent optics are prevalent. The latter is in many ways more of an art than a science because of the skill re-

quired to achieve good results, using methods which conceptually are almost trivial. Its main advantages are that vast amounts of information can be stored on a small area of film and processing is virtually instantaneous once lenses and filters have been correctly positioned on the optical bench. As an example, a coherent optical system at the University of Michigan is capable of processing 70mm films with a resolution of 100 cycles/mm which allows instant Fourier transformation of about $(2\times70\times100)^2 \simeq 2\times10^8$ points as opposed to several hours on a fast digital computer. On a small scale a coherent optical processing system costs considerably less than a high speed digital computer, but large high quality diffraction-limited (*i.e.* aberration free) systems are very expensive, the system at the University of Michigan costing $500,000. The foremost disadvantages are its inflexibility owing to the difficulty of fabricating filters for anything but the simplest processing applications and its limitation to linear space-invariant processing. Digital computation on the other hand has a great advantage in that systems may be arbitrarily defined and manipulated and processing is exactly reproducible. In addition once the software has been prepared, the system may be semi-automated so that a relatively untrained technician could operate the system since his only function would be the selection of processes and parameters and surveillance of I/O data.

Image processing using optical techniques had its origins in the experiments of Abbé and Porter, but it was not until the 1950's that the potential of spatial frequency filtering was recognized by Maréchal. The development of lasers has increased the power and flexibility of optical processing by providing a strong source of coherent radiation enabling interference techniques to be implemented with holographic filters.

Digital image processing has blossomed forth in the

last decade with the widespread availability of high speed digital computers with large fast memories. Classical filtering techniques inherited from optical processing have been successfully effected, and more recently optimum nonrecursive and recursive methods, which are specifically applicable to digital processing, have been introduced.

This thesis has been devoted almost exclusively to digital processing techniques since these appear to have the greatest scope and future. The second chapter deals with various unitary transforms and algorithms for eliminating redundant calculations in their computation since these are of considerable importance in image processing. The third and fourth chapters are concerned with restoration and enhancement techniques. The treatment has a theoretical bias, actual practical descriptions being confined mainly to the Appendices containing some of the computer programs used in processing images illustrating this work and brief notes on the microdensitometer used for digitizing and printing images.

TRANSFORMS CHAPTER 2

Transform techniques have two main applications in image processing. Firstly they can streamline and reduce the amount of calculation in a particular process, the most well-known example of which is probably convolving functions by taking the product of their spatial frequency spectra, using the Fast Fourier Transform algorithm of Cooley and Tukey[1] to calculate the necessary Discrete Fourier Transforms (DFT).Secondly they can be used to produce a more convenient distribution of the information in an image; for example by concentrating most of the information in one area, the Hadamard transform permits bandwidth reduction in image coding, and by resolving an image into its different spatial frequency components, the Fourier transform enables one to modify an image by adjusting the relative amplitudes of various spatial frequencies.

It is convenient to write an N-point transform as an N×N matrix. If the N×N transform matrix is denoted by T, and x and y are column vectors of length N, then y the T transform of x, is

$$y = Tx \qquad (2.1)$$

which is a one-dimensional transform. The extension to two dimensions for a separable transform with a symmetric kernel is trivial. (The Fourier, Hadamard and Haar transforms all satisfy this property, whereas the Karhunen-Loéve (or Hotelling) transform is not in general separable, though examples in which it is separable are known[2]).

$$\text{Let } Y' = \{y_1, y_2, \ldots, y_N\}$$
$$X = \{x_1, x_2, \ldots, x_N\} \qquad (2.2)$$

with $y_i' = Tx_i$.

Then clearly

$$Y' = TX$$

so that Y' is the column transform of the N×N matrix X. To perform the row transform of Y' we form the transpose, Y'^{T}, and determine its column transform:

$$\begin{aligned} Y^{T} &= TY'^{T} \\ &= T(TX)^{T} \end{aligned}$$

Transposing again, we have that the two-dimensional T transform of X is

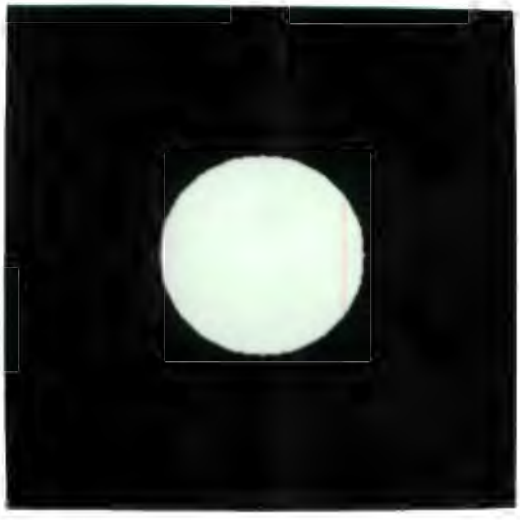
$$Y = TXT^{T} \qquad (2.3)$$

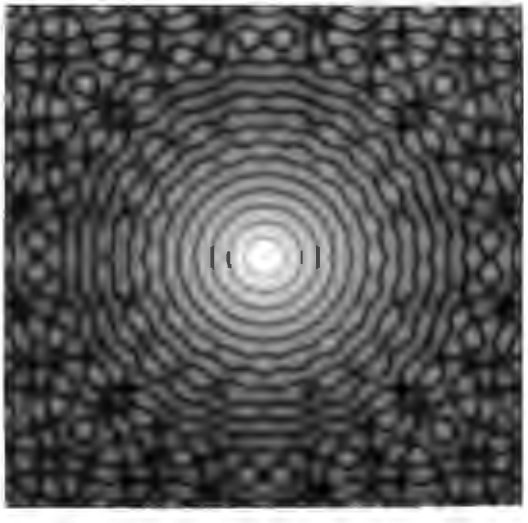
Hence to perform a two-dimensional transform one calculates the one-dimensional transforms of the rows and then the columns or *vice versa*.

Plate 1 shows two circles and their DFT's and Haar transforms (the arrays comprise 256×256 points). Plate 2 shows the Hadamard transforms of the same circles. (Note that the transforms are displayed on a logarithmic scale so that their high frequency components will be clearly visible. Plate 2 also shows the Hadamard transform of an array with the amplitude of the (1,1) component equal to two units and the (5,10) and (40,50) components each equal to one unit, the remainder being zero, and an optical Fourier transform which should be compared with the DFT of the small circle in Plate 1. The periodic nature of the DFT basis vectors implies that the arrays are continued periodically in all directions so that aliasing errors often occur, as will be seen in this case. The optical diffraction pattern is of a similar periodic nature in its centre, being the optical Fourier transform of a rectangular grid in a circular aperture.

Plate 1:

Top left: *circle, diamter 101.*
Top right: *circle, diameter 33.*
Middle: *DFT's of the top images.*
Bottom: *Haar transforms of the top images.*





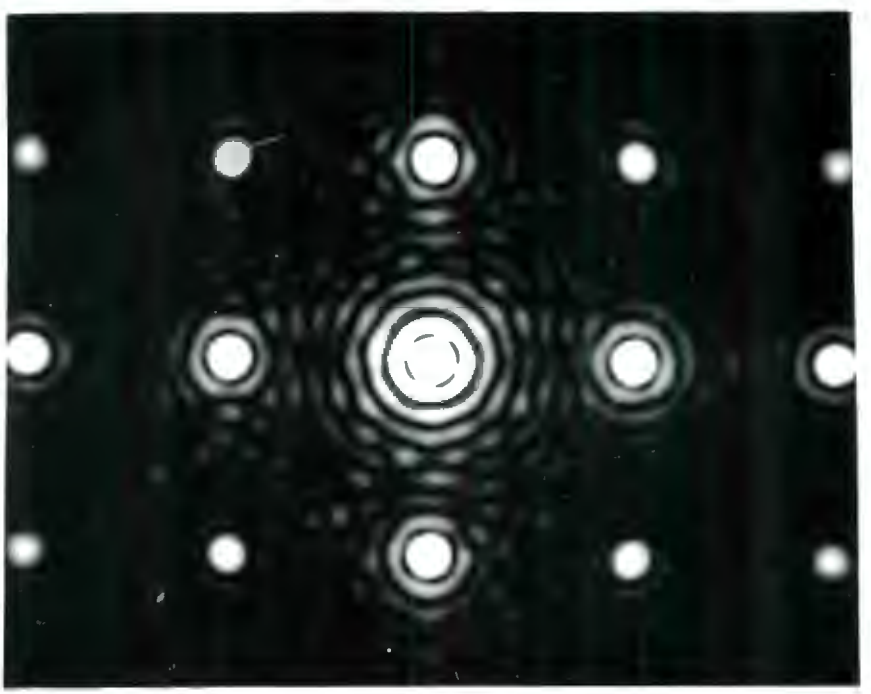


Plate 2:

Top: *Hadamard transforms of the circles of Pl. 1.*

Middle: *Hadamard transform of array with (1,1) element equal to 2 units, (5,10) and (40,50) elements equal to 1 unit and the remainder zero.*

Bottom: *Optical Fourier transform of a rectangular grid in a circular aperture.*

Each transform is naturally more suited to one application than another, so that a particular situation will dictate the use of one in preference to others. In general one has to compromise between optimization and computational efficiency. An illustration of this is in image coding where it is desired to optimize bandwidth reduction for given image quality: the Karhunen- Loéve transform is optimum but involves of the order of N^2 multiplications as opposed to $N\log_2 N$ multiply and add operations for the FFT and $N\log_2 N$ additions for the Fast Hadamard transform which calculate the Discrete Fourier and Hadamard transforms according to algorithms which avoid redundant calculations. In addition the Fourier and Hadamard transforms are invariant for different images, whereas the Karhunen-Loéve transform matrix is determined from the expectation value of the covariance matrices of the images to be processed, and thus will depend on their properties.

It is clear that a decrease by a factor of $N/\log_2 N$ in the amount of calculation required in processing an image will be of considerable importance for large image arrays. As a result much effort has been spent on developing algorithms for implementing these savings, the most celebrated of which is the FFT. The basic principle involved is that transform matrices possessing these algorithms may be factored into several sparse matrices (*i.e.* having few non-zero elements). A paper by Good[3] laid the basis for these fast algorithms. We shall outline the relevant points of his paper before considering specific algorithms.

Let the row suffix, r, of a matrix be written as

$$r = (r_1, r_2, \ldots, r_n),\ r_i = 0,1,\ldots,t-1;\ i = 1,2,\ldots,n,$$

with a similar representation for the column suffix, s. Then the direct (or Kronecker) product of n $t\times t$ matrices,

$M^{(i)}$, is defined as the $t^n \times t^n$ matrix A whose r,s^{th} element is

$$A_{r,s} = \prod_{i=1}^{n} M^{(i)}_{r_i, s_i} \qquad (2.4)$$

This definition may readily be extended to products in which the $M^{(i)}$ are not necessarily of the same order. It suffices to restrict the range of the r_i, s_i to $0,1,\ldots,t_i-1$; the resultant matrix has dimension $\tau = \Pi t_i$. As an example, consider the direct product of a 3×3 and a 2×2 matrix:

$$P = \begin{pmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & p_{22} \end{pmatrix} \qquad Q = \begin{pmatrix} q_{00} & q_{01} \\ q_{10} & q_{11} \end{pmatrix}$$

and

$$P \times Q = \begin{pmatrix} p_{00}q_{00} & p_{00}q_{01} & p_{01}q_{00} & p_{01}q_{01} & p_{02}q_{00} & p_{02}q_{01} \\ p_{00}q_{10} & p_{00}q_{11} & p_{01}q_{10} & p_{01}q_{11} & p_{02}q_{10} & p_{02}q_{11} \\ p_{10}q_{00} & p_{10}q_{01} & p_{11}q_{00} & p_{11}q_{01} & p_{12}q_{00} & p_{12}q_{01} \\ p_{10}q_{10} & p_{10}q_{11} & p_{11}q_{10} & p_{11}q_{11} & p_{12}q_{10} & p_{12}q_{11} \\ p_{20}q_{00} & p_{20}q_{01} & p_{21}q_{00} & p_{21}q_{01} & p_{22}q_{00} & p_{22}q_{01} \\ p_{20}q_{10} & p_{20}q_{11} & p_{21}q_{10} & p_{21}q_{11} & p_{22}q_{10} & p_{22}q_{11} \end{pmatrix}$$

Good's main result is that for any $t \times t$ matrix M

$$M^{[n]} = B^n \qquad (2.5)$$

where

$$B_{r,s} = M_{r_1, s_n} \delta^{s_1}_{r_2} \delta^{s_2}_{r_3} \ldots\ldots \delta^{s_{n-1}}_{r_n}$$

and $[n]$ indicates the n^{th} direct power ($M^{(i)} = M \;\forall\; i$ in the theorem); δ^i_j is the Kronecker delta: $\delta^i_j = 0$, $i \neq j$, $\delta^i_i = 1$. If the $M^{(i)}$ are of order t_i then the statement may be modified to

$$M^{(1)} \times M^{(2)} \times \ldots\ldots\ldots \times M^{(n)} = C^{(1)} C^{(2)} \ldots\ldots C^{(n)}$$

where

$$
\begin{aligned}
C^{(1)} &= M^{(1)} \times I_{t_2} \times \ldots\ldots\ldots\ldots \times I_{t_n} \\
C^{(2)} &= I_{t_1} \times M^{(2)} \times \ldots\ldots\ldots\ldots \times I_{t_n} \\
&\cdot \qquad\qquad\qquad\qquad\qquad\qquad \cdot \\
&\cdot \qquad\qquad\qquad\qquad\qquad\qquad \cdot \\
C^{(n)} &= I_{t_1} \times I_{t_2} \times \ldots\ldots\ldots\ldots \times M^{(n)}
\end{aligned}
$$

The matrices B and $C^{(i)}$ have at most t^{n+1} and $t_i \tau$ non-zero elements respectively, so that n applications of these matrices to a vector of length t^n or τ will require nt^{n+1} or $\tau \Sigma t_i$ multiplications instead of t^{2n} or τ^2 in the case of an unfactored matrix. Expressing this in more familiar notation with $N = t^n$ we have that $t N \log_t N$ will be needed compared with N^2 for an unfactored transform matrix. Clearly there is an enormous saving for large N.

Let us now apply Good's theorem to the Hadamard transform to obtain a factorization of the Hadamard matrix as a first step towards determining a Fast Hadamard Transform algorithm. A Hadamard matrix comprises 1's and -1's and has orthogonal rows and columns. Now a Hadamard matrix of order 2^n may be constructed by the direct product of the 2×2 Hadamard matrix with itself n times and consequently the factorization may be implemented directly from Good's theorem.

The unordered Hadamard transform matrix may be defined by

$$H_{r,s} = (-1)^{\sum_{i=0}^{n-1} r_i s_i} \qquad (2.6)$$

where r_i and s_i are the components of r and s as defined above. The submatrix is

$$M = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Thus for $n = 3$

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}$$

and

$$B = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

$$= \{M_{r_1, s_3} \delta^{s_1}_{r_2} \delta^{s_2}_{r_3}\}$$

so that

$$Hx = M^{[3]}x = B^3 x$$

We cannot write down the DFT equivalent directly, but must introduce a few modifications. Andrews[4] presents two examples of factorizations of DFT matrices with a product comprising alternately Good and diagonal matrices but gives no justification. However we can derive the form of this product fairly easily from the Cooley-Tukey formulation of the FFT.

Using the conventional notation (see for example Brigham[5] for a neat treatment of the FFT) that $W = e^{-j2\pi/N}$, $N = 2^n$ and writing in binary notation

$$k = 2^{n-1}k_{n-1}+2^{n-2}k_{n-2}+.....+k_0$$
$$l = 2^{n-1}l_{n-1}+2^{n-2}l_{n-2}+.....+l_0$$

we have that X the DFT of x is given by

$$X(k_{n-1},k_{n-2},\ldots,k_0) = \sum_{l_0=0}^{1}\sum_{l_1=0}^{1}\ldots\sum_{l_{n-1}=0}^{1} x(l_{n-1},l_{n-2},..,l_0)W^i$$

where $i = kl = (2^{n-1}k_{n-1}+\ldots+k_0)(2^{n-1}l_{n-1}+..+l_0)$

So

$$W^i = W^{k2^{n-1}l_{n-1}}W^{k2^{n-2}l_{n-2}}W^{k2^{n-3}l_{n-3}}.....W^{kl_0}$$

Now

$$W^{k2^{n-1}l_{n-1}} = W^{(2^{n-1}k_{n-1}+2^{n-2}k_{n-2}+.....+k_0)2^{n-1}l_{n-1}}$$
$$= [W^{2^n(2^{n-2}k_{n-1}l_{n-1})}][W^{2^n(2^{n-3}k_{n-2}l_{n-1})}]\ldots$$
$$.....[W^{2^n(k_1l_{n-1})}]W^{2^{n-1}(k_0l_{n-1})}$$
$$= W^{2^{n-1}(k_0l_{n-1})}$$

since $W^{2^n} = (e^{-j2\pi/N})^N = 1$.

Similarly the second factor $W^{k2^{n-2}l_{n-2}} = W^{(2k_1+k_0)2^{n-2}l_{n-2}}$.

Hence $X(k_{n-1},k_{n-2},\ldots,k_0)$

$$= \sum_{l_0}\sum_{l_1}\ldots\sum_{l_{n-1}} x(l_{n-1},l_{n-2},\ldots,l_0)W^{2^{n-1}k_0l_{n-1}}W^{2^{n-2}(2k_1+k_0)l_{n-2}}\ldots$$
$$\ldots W^{(2^{n-1}k_{n-1}+\ldots+k_0)l_0}$$

If the vector resulting from performing the j^{th} summation

is denoted by x_j, then we have

$$x_1(k_0, l_{n-2}, l_{n-3}, \ldots, l_0) = \sum_{l_{n-1}} x(l_{n-1}, \ldots, l_0) W^{2^{n-1} k_0 l_{n-1}}$$

$$x_2(k_0, k_1, l_{n-3}, \ldots, l_0) = \sum_{l_{n-2}} x_1(k_0, l_{n-2}, \ldots, l_0) W^{2^{n-2}(2k_1+k_0) l_{n-2}}$$

$$\vdots \qquad\qquad\qquad\qquad \vdots$$

$$x_n(k_0, k_1, \ldots, k_{n-1}) = \sum_{l_0} x_{n-1}(k_0, k_1, \ldots, l_0) W^{2^{n-1} k_{n-1} + \ldots + k_0) l_0}$$

and finally

$$X(k_{n-1}, k_{n-2}, \ldots, k_0) = x_n(k_0, k_1, \ldots, k_{n-1})$$

Note that the last line expresses the unscrambling of the transformed vector using the technique of bit reversal.

Now inspection of the exponents of the W's reveals that the first term may be represented by a Good matrix constructed from a submatrix

$$M_n = \begin{bmatrix} 1 & 1 \\ 1 & W^{2^{n-1}} \end{bmatrix}$$

while subsequent terms will result in certain columns of the Good matrices being multiplied by lower powers of W. This can be accomplished by post-multiplication of the Good matrices by diagonal matrices with suitable elements. Now the first summation has only one term in the exponent, so that the first Good matrix, G_0, requires no modification. The second summation contains the additional term $2^{n-2} k_0 l_{n-2}$ which will result in colums whose binary representations have 1's as the first and second bits being multiplied by $W^{2^{n-2}}$. This may be accomplished by post-multiplying the second Good matrix G_1 by a diagonal matrix D_1 with the last 2^{n-2} equal to $W^{2^{n-2}}$ and the remainder unity. The third summation has a modification given by $2^{n-3}(2k_1+k_0) l_{n-3}$

so that columns with first and third bits equal to 1 should be multiplied by $W^{2^{n-3}}$ (because of $2^{n-3}k_0 l_{n-3}$) and columns with second and third bits unity are multiplied by $W^{2^{n-2}}$. This is accomplished by placing these values in the appropriate diagonal elements of D_2. The process may be continued similarly for the remainder of the matrices.

All that remains now is to determine the method of constructing the Good matrices from the submatrix M. We see from the exponents of the W's and the subscripts of the data vectors that during the j^{th} summation vector elements separated by 2^{n-j} are combined so that the Good matrices are given by

$$
\begin{aligned}
G_0 &= M_n \times I \times \ldots\ldots\ldots \times I \\
G_1 &= I \times M_n \times \ldots\ldots\ldots \times I \\
&\vdots \qquad\qquad \vdots \\
G_{n-1} &= I \times I \times \ldots\ldots \times I \times M_n
\end{aligned}
$$

where

$$
I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}
$$

Then the Fourier transform may be written

$$
F = G_{n-1} D_{n-1} \ldots\ldots G_1 D_1 G_0
$$

Note that the G_i should be combined with the D_i once the factorization has been determined.

We shall illustrate this process with an 8-point DFT so that $n = \log_2 8 = 3$.

$$
M_3 = \begin{pmatrix} 1 & 1 \\ 1 & W^4 \end{pmatrix}
$$

$$G_0 = \begin{pmatrix} 1 & 1 \\ 1 & W^4 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & W^4 & 0 \\ 0 & 1 & 0 & W^4 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & W^4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W^4 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W^4 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W^4 \end{pmatrix}$$

$$G_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 \\ 1 & W^4 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & W^4 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & W^4 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & W^4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W^4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & W^4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W^4 \end{pmatrix}$$

Similarly $G_2 = I \times I \times M_3$

$$\text{and} \quad G_2 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & W^4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W^4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W^4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W^4 \end{pmatrix}$$

$$D_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & W^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^2 \end{pmatrix}$$

$$D_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & W^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & W^1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^3 \end{pmatrix}$$

$$\text{and} \quad F = \begin{pmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^4 & W^0 & W^4 & W^0 & W^4 & W^0 & W^4 \\ W^0 & W^2 & W^4 & W^6 & W^0 & W^2 & W^4 & W^6 \\ W^0 & W^6 & W^4 & W^2 & W^0 & W^6 & W^4 & W^2 \\ W^0 & W^1 & W^2 & W^3 & W^4 & W^5 & W^6 & W^7 \\ W^0 & W^5 & W^2 & W^7 & W^4 & W^1 & W^6 & W^3 \\ W^0 & W^3 & W^6 & W^1 & W^4 & W^7 & W^2 & W^5 \\ W^0 & W^7 & W^6 & W^5 & W^4 & W^3 & W^2 & W^1 \end{pmatrix} = G_2 D_2 G_1 D_1 G_0$$

Note that the diagonal matrices given by Andrews differ

from those derived by the method given above, but that they are equivalent.

The factored form of the Haar[6] matrix (see Appendix a for a definition of Haar functions) has certain similarities to the Hadamard decomposition but is not directly implementable from Good's theorem. Andrews[4] gives the factorization for N = 8: (putting $\sqrt{2} = \rho$)

$$H_1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

$$H_2 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \rho & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \rho & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \rho & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho \end{pmatrix}$$

$$H_3 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \rho & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \rho & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \rho & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \rho & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \rho & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho \end{pmatrix}$$

and $H = H_3 H_2 H_1$

so that H =

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \rho & \rho & -\rho & -\rho & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \rho & \rho & -\rho & -\rho \\ \rho^2 & -\rho^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \rho^2 & -\rho^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \rho^2 & -\rho^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \rho^2 & -\rho^2 \end{pmatrix}$$

We can see however that the i^{th} matrix comprises a sub-matrix E_i inserted into the top left of a diagonal matrix comprising entries of $\sqrt{2}$, where

$$E_i = \{M_{r_1, s_i} \delta^{s_1}_{r_2} \delta^{s_2}_{r_3} .. \delta^{s_{i-1}}_{r_i}\}$$

Inspection of the matrices shows that the factorization allows the transform to be performed by 2(N-1) additions and $N\log_2 N - 2(N-1)$ multiplications by $\sqrt{2}$ (Andrews omits the latter requirement).

Another transform possessing a fast computation algorithm is the Slant transform[7] which was recently designed specifically for image coding. The basis vectors for this transform are staircases of various lengths, as well as functions of similar appearance to Walsh functions, though some are multilevel. The staircase vectors were chosen to match gradual intensity changes in an image. The more closely the basis vectors match an image, the greater will be the information compaction in the transform domain, which is the reason for the optimum performance of the Karhunen-Loéve transform. The Slant transform has been found to result in a lower mean square error for moderate sized image blocks (say up to 32×32) than Fourier, Hadamard and Haar transforms for given bandwidth reduction. The performance of the Karhunen-Loéve transform is superior but it suffers from the disadvantage of not having a fast computation

algorithm.

The transform may be generated by the following recursive relations[7]:

$$S_N = \frac{1}{\sqrt{2}} \begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & b_N & 0 & a_N & 0 & 0 \\ 0 & 0 & I_{(N/2)-2} & 0 & 0 & 0 \\ 0 & a_N & 0 & -b_N & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{N/2-2} \end{bmatrix} \begin{bmatrix} I_{N/2} & I_{N/2} \\ I_{N/2} & -I_{N/2} \end{bmatrix} \begin{bmatrix} S_{N/2} & 0 \\ 0 & S_{N/2} \end{bmatrix} \end{bmatrix}$$

$$S_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$a_2 = 1$$

$$b_N = [1 + 4(a_{N/2})^2]^{-\frac{1}{2}}$$

$$a_N = 2b_N a_{N/2}$$

or

$$a_{2N} = \left(\frac{3N^2}{4N^2-1}\right)^{\frac{1}{2}}$$

$$b_{2N} = \left(\frac{N^2-1}{4N^2-1}\right)^{\frac{1}{2}}$$

where I_n is the $n \times n$ identity matrix.

A total of $N\log_2 N + N/2 - 2$ additions and $2N - 4$ multiplications are required to compute the Slant transform using its fast algorithm.

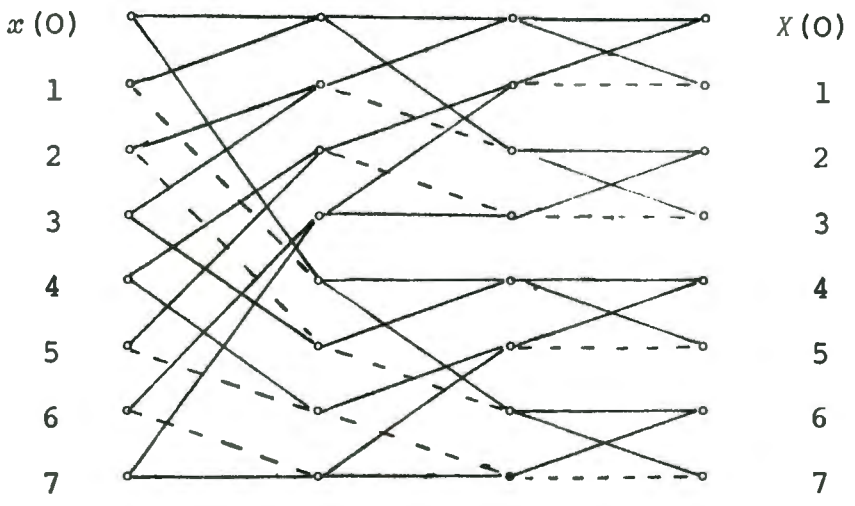
We are now in a position to determine fast computation algorithms for the transforms mentioned above.

The flow diagrams defining algorithms for fast transformations may be written down directly from the matrix factorization. We shall not dwell on the FFT algorithm as it has been extensively documented.[8] A subroutine implementing the FFT written by N. Brenner of MIT Lincoln Lab. is given in the Appendix. The two commonest forms are the Cooley-Tukey and Sande-Tukey versions. The algorithms for both may be written either to produce an unordered transformed vector from an ordered input vector or an ordered output vector from a pre-shuffled input vector. In both cases shuffling is achieved by interchanging elements whose binary location representations are bit-reversals of each other. The FFT subroutine given in the Appendix is of the latter type.

Let us consider the Fast Hadamard Transform. Whereas the DFT is ordered in frequency, the HFT may be ordered in *sequency*, a term introduced by Harmuth[9]. Sequency is defined as the number of transitions from 1 to -1 and *vice versa* in a row of a Hadamard matrix. The Hadamard matrices of order 2^n constructed by the direct product method outlined above possess all sequencies from 0 through 2^n-1. However they are not ordered in sequency. Thus to obtain an algorithm which produces an ordered output vector directly, without requiring a post-transform shuffle, it is necessary to rearrange the elements of the factored matrices, with the result that the individual matrices are no longer identical. The matrices for N = 8 after rearrangement are:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

From this we may write down the flow graph for N = 8 by inspection, where solid lines indicate addition and dashed lines subtraction:



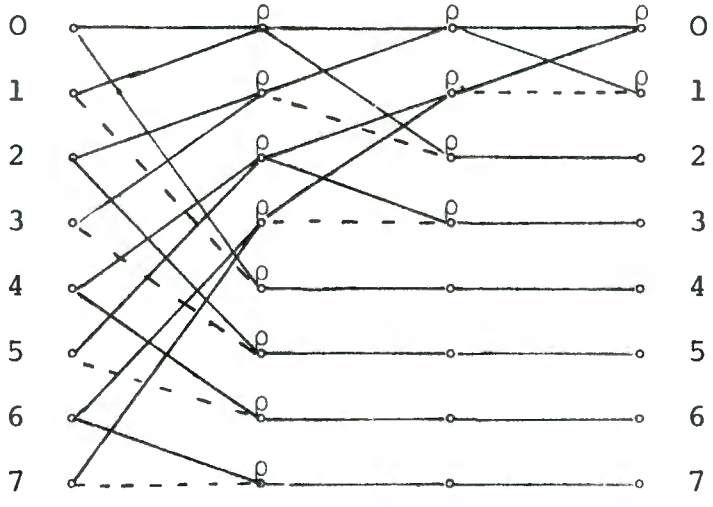
Pratt *et al.*[10] have presented an algorithm implementing this graph by which calculations are performed according to a sieving sequence. This seems unnecessarily complicated since inspection of the flow graph reveals a much simpler rule. Clearly for an N point transform, in evaluating the first transition, the first N/2 values are sums of sequential pairs while the other N/2 are differences. For the second transition, the first N/4 values are sums of sequential pairs followed by N/4 differences, this set of operations being performed twice, and for the i^{th} there are $N/2^i$ sums, followed by $N/2^i$ differences, each set repeated 2^{i-1} times. This is readily implemented with *DO-loops* in a computer program, and is the method used in the FHT subroutine given in the Appendix. As the Hadamard matrix is its own inverse (except for a scaling factor of $\sqrt{N}$) the forward and inverse transforms are performed by the same program.

The flow graph for the Fast Haar Transform is similar, though multiplications by $\sqrt{2}$ are also required. The Haar matrix given above is not normalized but division by $\sqrt{N}$ will accomplish this, and this may

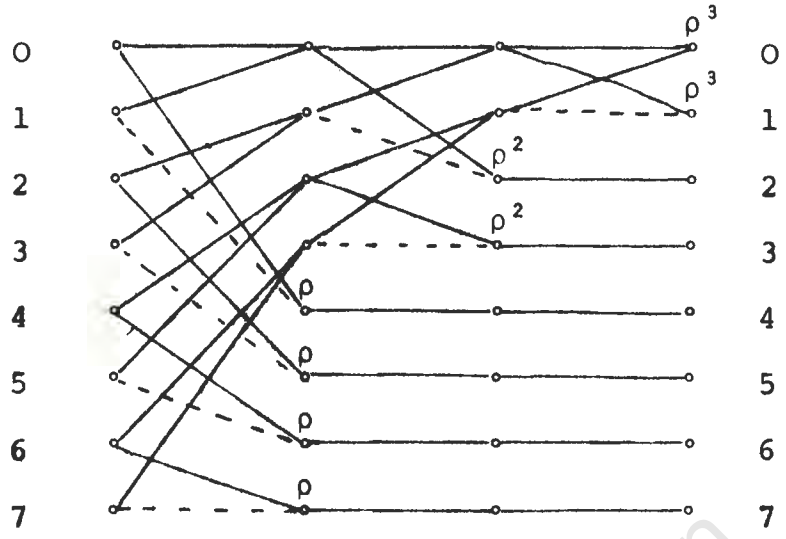
be done in practice by dividing each product matrix by $\sqrt{2}$ since $N = 2^{\log_2 N}$. For N = 8 the normalized product is thus (putting $\rho = 1/\sqrt{2}$):

$$\begin{bmatrix} \rho & \rho & 0 & 0 & 0 & 0 & 0 & 0 \\ \rho & -\rho & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \rho & \rho & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \rho & \rho & 0 & 0 & 0 & 0 \\ \rho & -\rho & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \rho & -\rho & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \rho & \rho & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \rho & \rho & 0 & 0 & 0 & 0 \\ 0 & 0 & \rho & -\rho & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \rho & \rho & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \rho & \rho \\ \rho & -\rho & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \rho & -\rho & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \rho & -\rho & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \rho & -\rho \end{bmatrix}$$

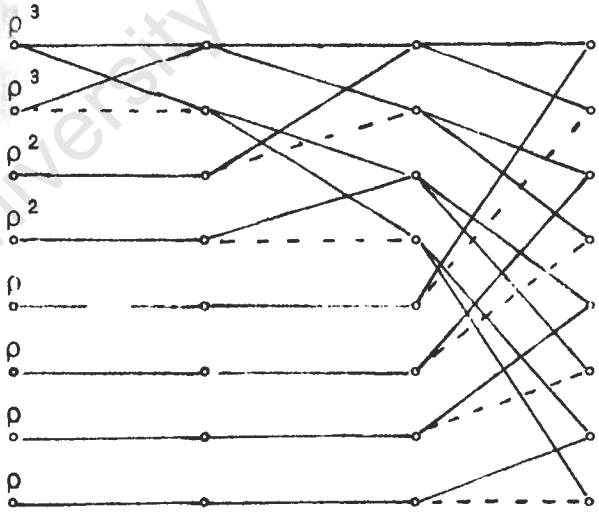
and the flow graph is



From this we see that at each node where a summing operation occurs, division by $\sqrt{2}$ is required. Clearly a computational saving will result if we divide only the *last* summing node for each element by $\sqrt{2^i}$ where the last node occurs at the i^{th} transition. The modified flow graph is thus as shown at the top of the next page, and is implemented by a subroutine in the Appendix. As a result, only 2(N - 1) additions and N multiplications are required.



The flow graph for the inverse transform may be written down from the factored inverted matrix of the forward transform, which in practice involves transposing the product matrices and reversing their sequence of application. The same reduction in multiplications may be achieved by premultiplying by the appropriate power of $\sqrt{2}$, and we shall present it in this form:



A subroutine performing the inverse Haar Transform is given in the Appendix.

A Hadamard transform of a Picker Thyroid phantom and a Haar transform of a pillbox distribution are shown in Pl. 24 for interest. The latter (Pl.24d) may be compared with the Haar transform shown in Pl. 1, where the image sampling interval is decreased by a factor of four, giving higher resolution.

IMAGE RESTORATION

CHAPTER 3

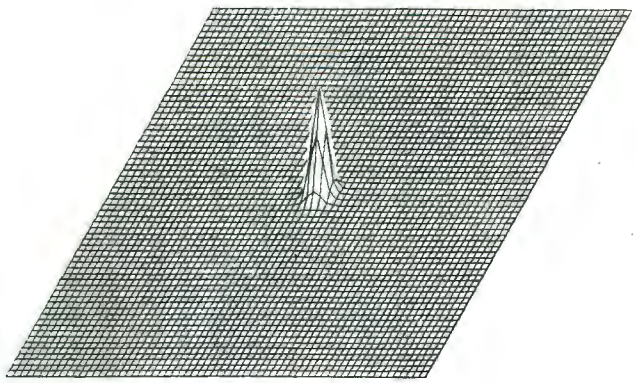
This is a broad topic and could be split into several sections. However it is difficult to draw dividing lines between the different areas as they overlap to such an extent; thus it seems preferable to treat the subject as a whole.

As was stated briefly in the introductory chapter, image restoration is concerned with identifying and characterizing the degradations which an image has undergone (*e.g.* addition of noise, geometric distortion) and then applying an optimum compensation process to recover a best estimate of the pure uncontaminated image, where *best* is subject to some criterion, such as minimization of the mean square error between the ideal image and the estimate.

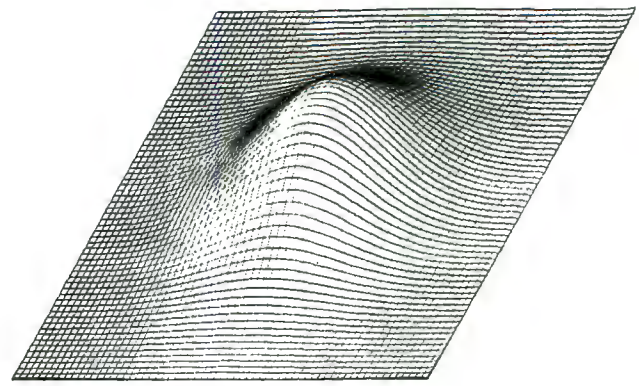
The degradation of an image is usually assumed to be linear, both because it facilitates analysis and design of compensation processes and also because the approximation is good. A linear degradation may be described by

$$g(u,v) = \int_{-\infty}^{\infty}\int f(x,y)h(u,v,x,y)\,dx\,dy + n(u,v) \qquad (3.1)$$

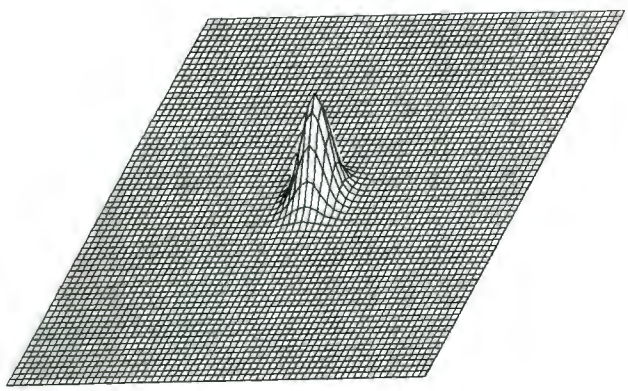
where f is the pure image, g the corrupted image, h the degrading function, known as the impulse response or point spread function and n is random noise, assumed to be additive. We see that in general h is a function of its position of application to the pure image; in other words different areas of the image are degraded in different ways. In this case the impulse response is said to be *space-variant*. However in many cases the impulse response remains almost unchanged for all image points and depends only on the difference between



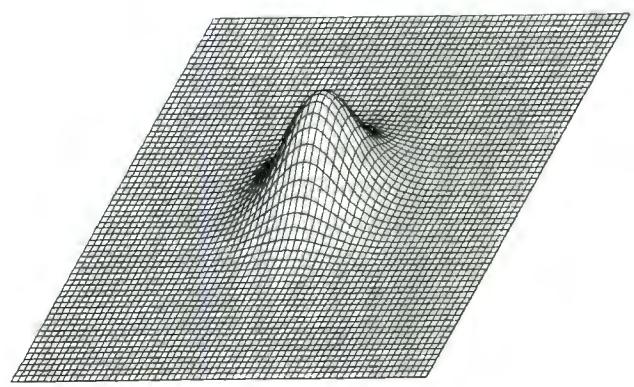
(A) GAUSSIAN FUNCTION ($E^{-R^2/2}$)



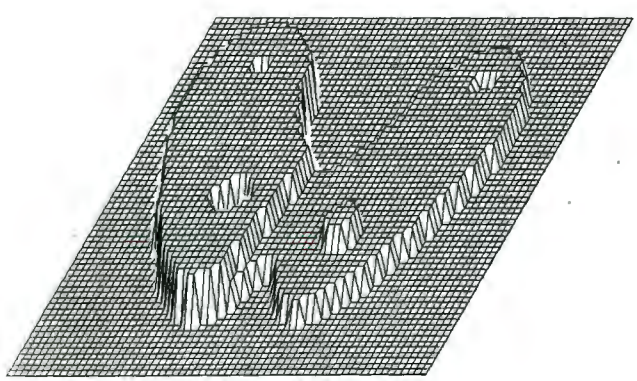
(B) DISCRETE FOURIER TRANSFORM OF (A)



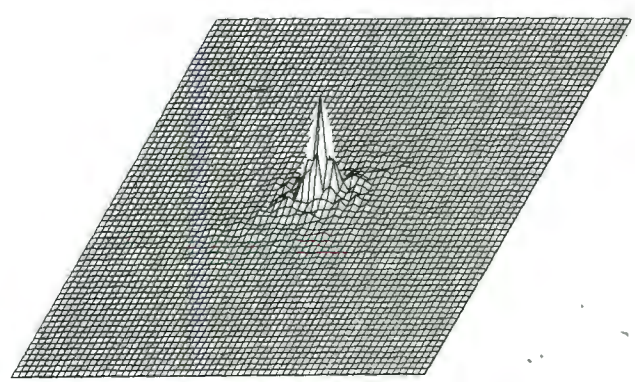
(C) WIDER GAUSSIAN FUNCTION ($E^{-R^2/8}$)



(D) DISCRETE FOURIER TRANSFORM OF (C)



(E) SIMULATED PICKER THYROID DISTRIBUTION



(F) DISCRETE FOURIER TRANSFORM OF (E)

the object and image coordinates. It is then said to be *space-invariant* and the impulse response simplifies to

$$h(u,v,x,y) = h(u - x, v - y) \qquad (3.2)$$

so that g may be expressed as a convolution

$$g(u,v) = f(u,v)*h(u,v) + n(u,v) \qquad (3.3)$$

Denoting Fourier transformed functions by upper case letters and using the fact that the Fourier transform of a convolution of two functions equals the product of their individual Fourier transforms, we have

$$G = F.H + N \qquad (3.4)$$

We shall first consider some simple examples of linear filtering before examining various restoration techniques which attempt to invert the effects of the impulse response, h.

A problem in almost every image processing technique is the presence of random fluctuations superimposed on the image and generally referred to as *noise*. In most cases we find that the spatial frequency spectrum (*i.e.* Fourier transform) of the noise is substantially flat for all frequencies. By this we mean that the spectral component amplitudes fluctuate within a constant range for all frequencies, though the individual amplitudes are certainly not constant, as is the case for white noise. Now in general the transform of any image has substantially low frequency content (see the simulated thyroid distribution and its DFT in Pl. 3e&f and the less detail there is in an image the faster its spectrum decreases with frequency. Thus given a noisy image g,

$$g = s + n \qquad (3.5)$$

we can obtain a better approximation to s by attenuating the high frequency components of g, while leaving low frequencies more or less unaltered, since by doing this we are removing a considerable fraction of the high frequency noise (which is generally much more offensive as regards the subjective image quality for a human viewer) while leaving the image information relatively unchanged. Clearly however one has to compromise between smoothing (noise reduction) and decreasing resolution (reducing high frequency components of the image). Furthermore, in this connection it will obviously be preferable in general to use a smoothing function which decreases smoothly to zero with increasing frequency rather than a pillbox, say, which will result in large ripple being introduced into the reconstructed image, and also because one is completely discarding the higher frequency information instead of reducing its presence.

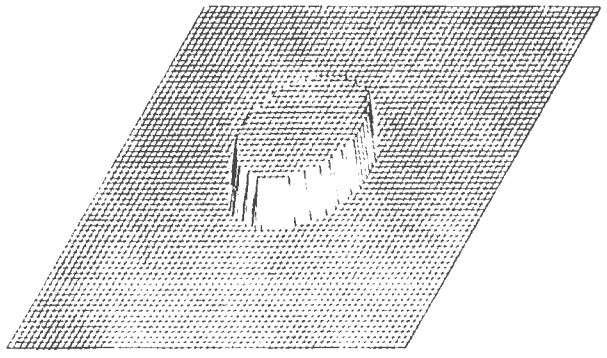
A satisfactory smoothing function for many purposes is a two-dimensional Gaussian function

$$e^{-(x^2+y^2)/a} = e^{-r^2/a}$$

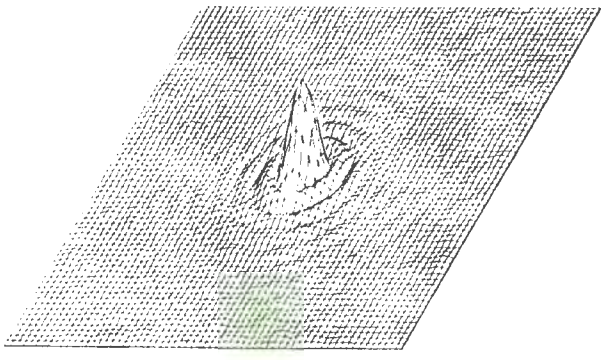
whose transform

$$e^{-a\pi^2 f_r^2}$$

is also a Gaussian function and thus does not introduce any ripple. The widths of the transform pair are clearly inversely proportional to one another (Pl. 3a-d). We may illustrate the use of a Gaussian smoothing function with a pillbox to which noise has been added. (See Pl. 4). The pillbox and its spectrum are shown (Pl. 4a&b), as well as with the addition of noise linearly distributed in the range $(-\frac{1}{2}, \frac{1}{2})$ where the pillbox has unit height



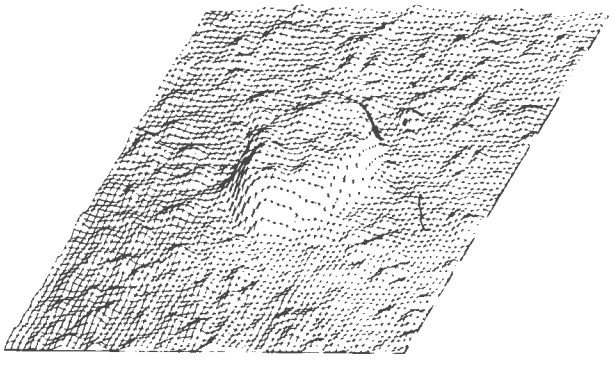
(A) PILLBOX DISTRIBUTION



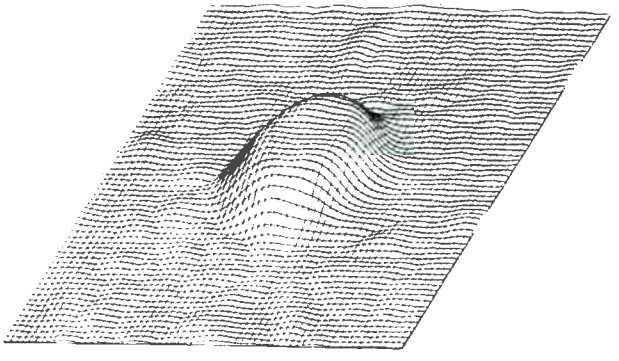
(B) DFT OF (A)

(C) UNIT HEIGHT PILLBOX WITH ADDITIVE NOISE IN RANGE (-1/2,1/2)

(D) DFT OF (C)



(E) RESULT OF SMOOTHING (C) WITH NARROW GAUSSIAN FUNCTION (PLATE 2(A)



(F) RESULT OF SMOOTHING (C) WITH WIDE GAUSSIAN FUNCTION (PLATE 2(C))

(Pl. 4c&d). The results of smoothing with the two Gaussian functions of Pl. 3 are shown.

If one does have to truncate an image spectrum abruptly for some reason, a *window* may be applied to the remaining nonzero components to reduce the ripple amplitude introduced into the image. Examples of such windows[11] are a *cosine bell*,

$$\cos(2\pi f/f_s)$$

where f_s is the sampling frequency of the image (the reciprocal of the sampling interval), the *Hanning window*,

$$\tfrac{1}{2}(1 + \cos(4\pi f/f_s))$$

and a slight modification, the *Hamming window*,

$$0{,}46 + 0{,}54\cos(4\pi f/f_s).$$

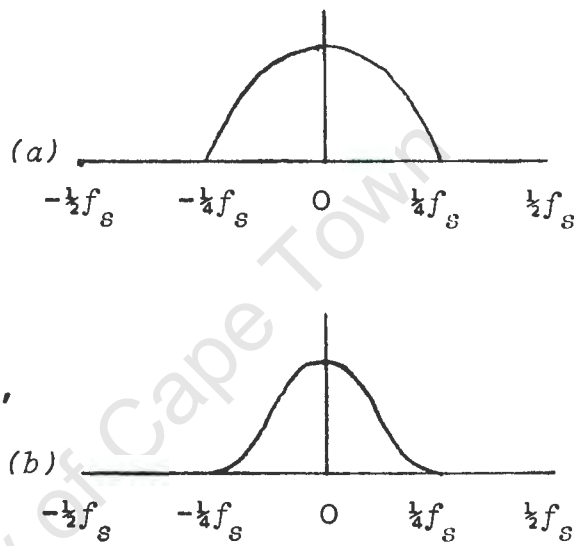


Fig. 1: (a) Cosine bell; (b) Hanning window.

There are many others but they have similar effects and in general decrease monotonically to zero over the truncated portions of the array. While these smoothing techniques are adequate in many cases, it is obvious that they cannot provide an optimum estimate for all images since they take no account of the statistical properties of images which clearly must vary considerably. Wiener[12] derived the form of the *linear* filter which will minimize the mean square error of the optimal estimate, $\hat{s}$, of s where

$$g = s + n \qquad (3.5)$$

We shall outline its derivation (see for example Davenport and Root[13]) in the one-dimensional case for simplicity. Extension to two dimensions is trivial.

Thus given an image, g, which is a corrupted form

of the pure image, s, we want to determine a linear filter r in terms of the power spectra of the signal, s, and noise, n, which will minimize the mean square difference between s and $\hat{s}$, *i.e.* we must minimize (note: all integrals are $\int_{-\infty}^{\infty}$ unless otherwise stated)

$$\int [s(x) - \int r(u)\, g(x-u)\, du]^2 dx \qquad (3.6)$$

We use a Lagrangian technique: since r is optimum the error will not be decreased if we replace r by $r+\lambda p$ where λ and p are arbitrary.

Hence

$$\int [s(x) - \int (r(u)+\lambda p(u))\, g(x-u)\, du]^2 dx - \int [s(x) - \int r(u)\, g(x-u)\, du]^2 dx \geq 0$$

Expanding and cancelling we find

$$2\lambda \{\iint r(u)\, g(x-u)\, du \int p(v)\, g(x-v)\, dv dx - \int s(x) \int p(u)\, g(x-u)\, du dx\}$$
$$+ \lambda^2 \iint p(u)\, g(x-u)\, du \int p(v)\, g(x-v)\, dv dx \geq 0$$

Now the second expression is nonnegative since the integrand is the square of a real function, *viz.*

$$\int p(u)\, g(x-u)\, du$$

Furthermore if the expression in braces is nonzero, by a suitable choice of λ small and positive or negative, the inequality will be violated.

Thus we require

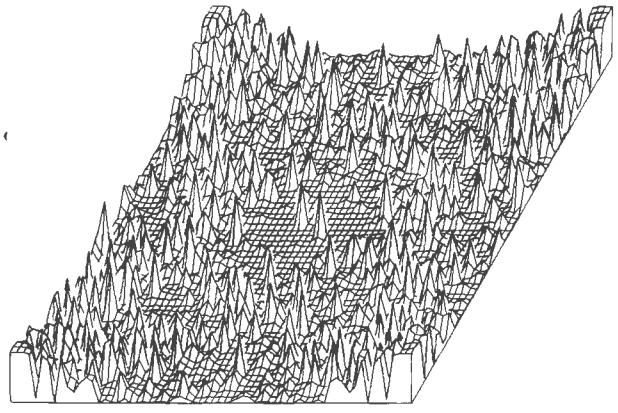
$$\int s(x) \int p(u)\, g(x-u)\, du dx = \iint r(u)\, g(x-u)\, du \int p(v)\, g(x-v)\, dv dx$$

which on changing the order of integration becomes

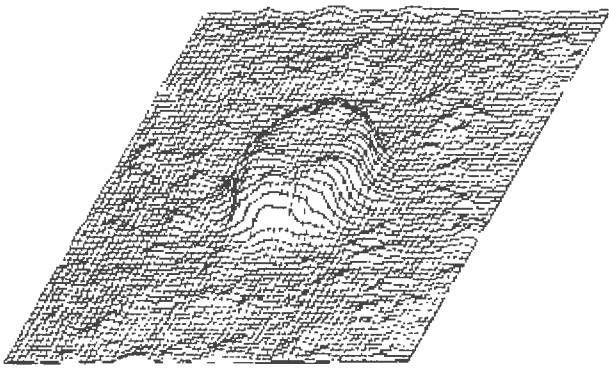
$$\int p(u) \int s(x)\, g(x-u)\, dx du = \iint r(u)\, p(v) \int g(x-u)\, g(x-v)\, dx dv du$$

and

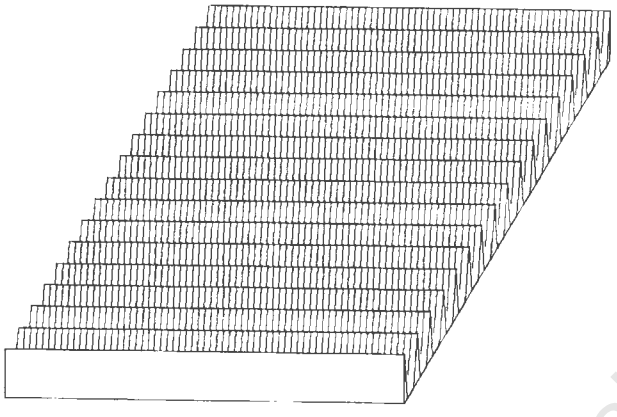
$$\int p(u) \left[\int s(x)\, g(x-u)\, dx - \int r(v) \int g(x-u)\, g(x-v)\, dx dv\right] du = 0$$



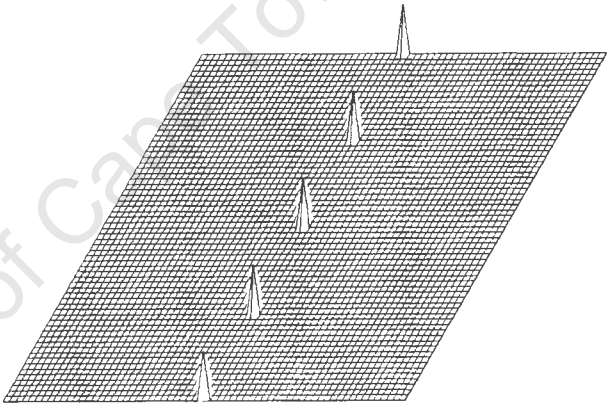
(A) PERFECT WIENER FILTER FOR RESTORING NOISY PILLBOX (PLATE 4(C))



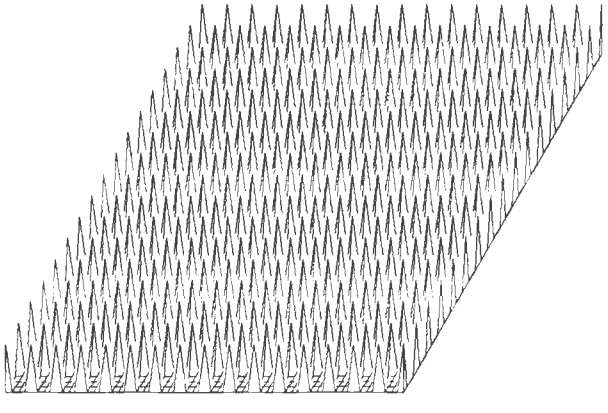
(B) RESULT OF WIENER FILTERING NOISY PILLBOX



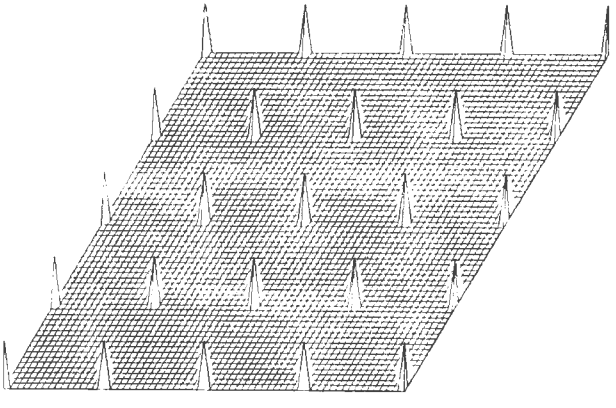
(C) FUNCTIONS FOR SAMPLING EVERY 4TH LINE OF AN ARRAY



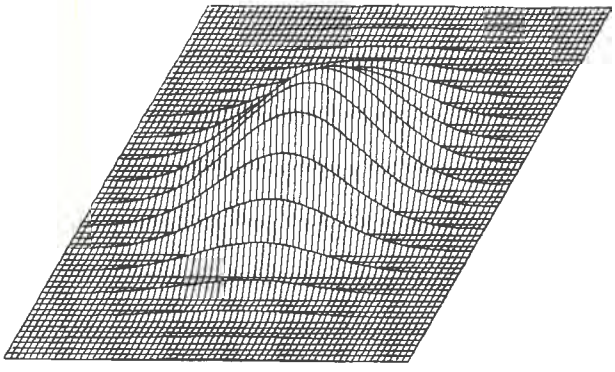
(D) DFT OF (C)



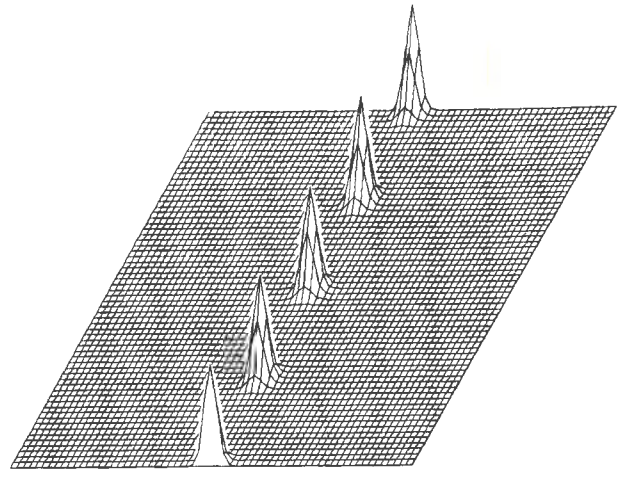
(E) FUNCTION FOR POINT-SAMPLING AN ARRAY



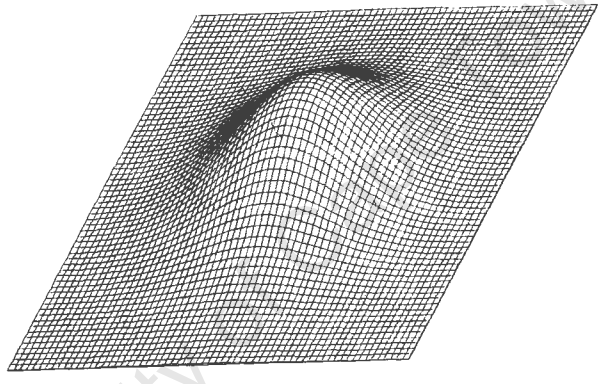
(F) DFT OF (E)



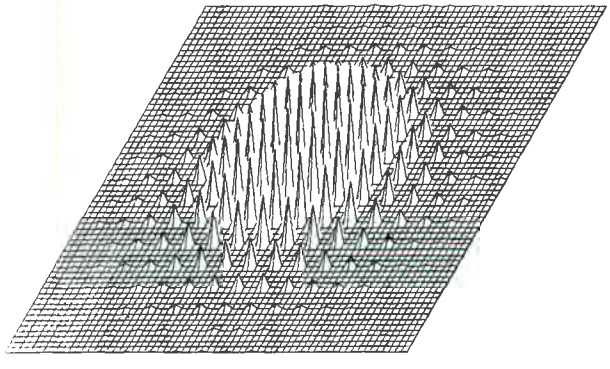
(A) RESULT OF LINE-SAMPLING A GAUSSIAN FUNCTION (PLATE 3B)



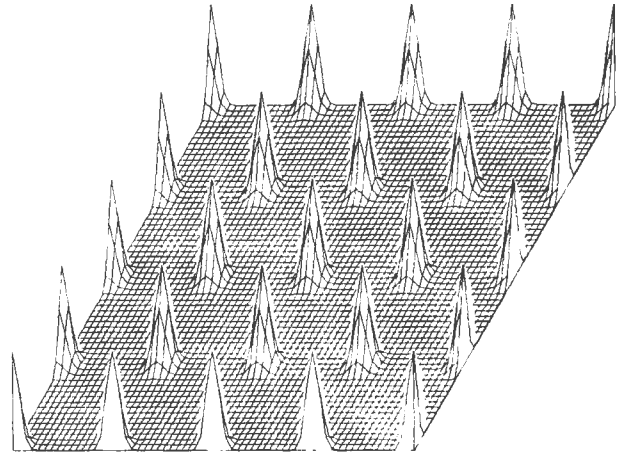
(B) DFT OF (A)



(C) RECONSTRUCTION OF ORIGINAL BY ZEROING ALL EXCEPT ONE SPECTRAL PATCH



(D) POINT-SAMPLED GAUSSIAN FUNCTION



(E) DFT OF (D)

Since p is arbitrary, we require the expression in brackets to be identically zero for the equation to be satisfied in general.

Hence

$$\int s(x)g(x-u)\,dx = \int r(v)\int g(x-u)g(x-v)\,dx\,dv$$

Fourier transforming, we get

$$S(f)G^{*}(f) = R(f)\,|G(f)|^{2}$$

so that

$$R = \frac{SG^{*}}{|G|^{2}} = \frac{S(S+N)^{*}}{(S+N)(S+N)^{*}}$$

Now, assuming s and n are uncorrelated and n has zero mean, we have $SN^{*} = S^{*}N = 0$ (an image s cannot have zero mean since it is nonnegative; the case $s \equiv 0$ everywhere is of no interest) and

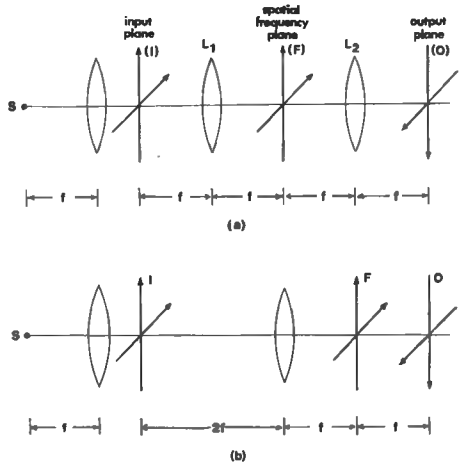
$$R = \frac{|S|^{2}}{|S|^{2}+|N|^{2}} = \frac{\Phi_S}{\Phi_S+\Phi_N} \qquad (3.7)$$

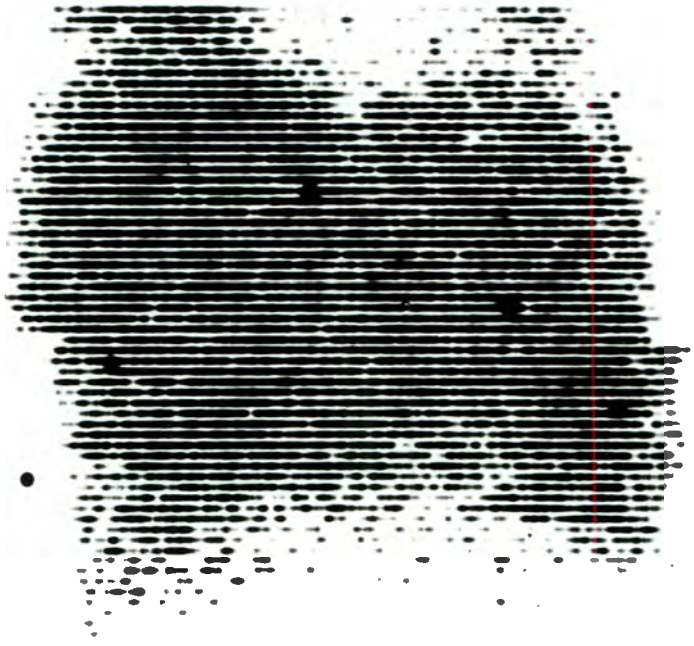
where Φ_S and Φ_N are image and noise power spectra. This is the Wiener filter for recovering a best mean square estimate of a signal contaminated with *additive* noise.

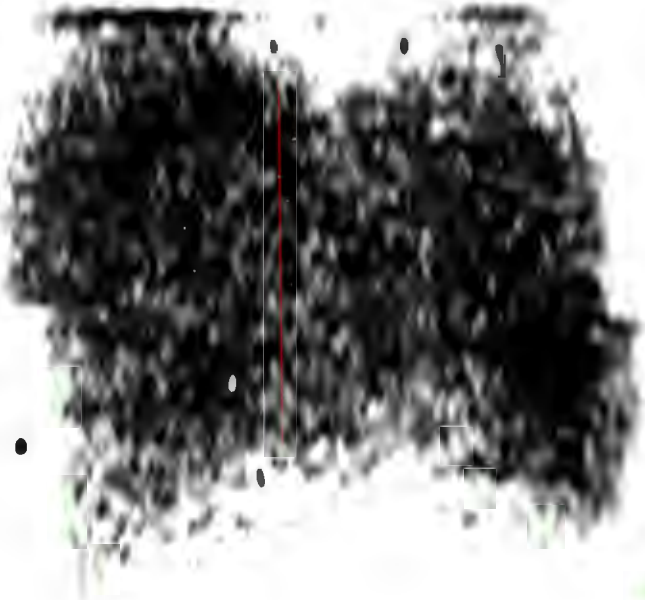
The Wiener filter for the noise-contaminated pill-box (Pl. 4c) is shown in Pl. 5a, together with the Wiener filtered reconstruction (Pl. 5b). It is clear that there is an improvement over the images (Pl. 4e&f) smoothed with Gaussian functions. Note that the Wiener filter shown is calculated from the actual power spectra of the pure image s and noise n, so that the minimum error bound is attained. In practice one would have to make assumptions about the spectral shapes *e.g.* that the noise spectrum is flat and the signal spectrum falls off as $\alpha^2/(\alpha^2+f^2)$ say.

Plate 7:

Top: *two coherent optical processing systems.*
Middle: *rectilinearly scanned liver and its optical Fourier transform*
Bottom: *derastered image obtained by removing all except the bright central patch of the spectrum.*


input plane
(I)
L_1
spatial frequency plane
(F)
L_2
output plane
(O)
S
f
f
f
f
f
(a)
I
F
O
S
f
2f
f
f
(b)





Another application of linear filtering of the Fourier spectrum of an image is interpolation. Schafer and Rabiner[14] have shown how to alter the separation of sampling points in an image. If an image is sampled at intervals of Δx then this is equivalent to multiplying the continuous image by an array of delta functions separated from each other by Δx. Hence the spectrum of the sampled image will be a convolution of the spectrum of the continuous image with the transformed delta array, which effectively results in the spectrum being repeated at intervals of $f_s = 1/\Delta x$. The only requirement is that the sampling interval should be fine enough (the Nyquist sampling criterion) to ensure that the spectral patches do not overlap, which would cause aliasing. From this we can see the principle of the technique: since the sampling interval is inversely proportional to the spectral patch separation, we can make the sampling intervals as small as we wish (*i.e.* interpolate) by increasing the spectral patch separation provided that the original image was adequately sampled.

Line- and point-sampling functions are shown together with their DFT's (Pl. 5c-f). The application of these sampling functions to the "continuous" image of Pl. 3b is shown in Pl. 6a&d, together with the reconstructed "continuous" image (Pl. 6c) obtained by zeroing all except one of the spectral patches (Pl. 6b&e), which it will be noticed are identical with that shown in Pl. 3a. The same process may be performed with coherent optics, since the image distribution in the spatial frequency plane of the first lens (see Pl. 7) is approximately the Fourier transform of the image in the input plane. A liver scan, obtained from a rectilinear scanner, and its spectrum are shown, together with the continuous image formed by filtering out all except the central bright

Plate 8:

Three other examples of liver scans which have been derastered in a coherent optical system in the same way as that shown in Pl. 7.

spectral patch of the periodic spectrum. Note that the irregularities in the image due to cirrhosis are much more readily apparent in the continuous image. Pl. 8 shows other examples of derastered liver scans. In practice the second optical system of Pl. 7 is preferable to the first since lenses invariably have defects so that the fewer used the better as coherent optical systems are very sensitive to phase variations along the optical path. Furthermore, in practice a small filtering aperture in the spatial frequency plane tends to smear out interference patterns introduced by defects in the front lenses, so that they are less offensive. Thus it is advantageous to position all lenses in front of the filter.

Schafer and Rabiner's analysis is similar to the method illustrated in Pl. 6: the interpolated points are obtained by multiplying the spectrum by a linear filter which removes all except one spectral patch, *i.e.* the "sampled" image is low-pass filtered. However we may gain more insight into the process and obtain a computationally more efficient method of implementing the interpolation by a direct analysis. We shall consider the case of halving the sampling intervals.

Given an N point array $x(j), j=0,1,\ldots,N-1$, we want to find a 2N point array $\hat{x}(j), j=0,1,\ldots 2N-1$, which is the interpolated version of x. Since an image is real we shall consider only the case of x real. Now the DFT of a real array has a spectrum with the property that

$$X(j) = X^{*}(N-j) \qquad j=1,2,\ldots,\tfrac{1}{2}N-1 \quad (3.8)$$

which corresponds to the result for the continuous Fourier transform. Hence when we construct the modified spectrum by doubling the spectral patch separation, we must ensure that this condition is still satisfied if we are to recover a real array.

We construct the modified spectrum as follows:

$$\begin{aligned}
\hat{X}(k) &= X(k) && k=0,..,\tfrac{1}{2}N-1\\
\hat{X}(N/2) &= \tfrac{1}{2}X(N/2)\\
\hat{X}(k) &= 0 && k=\tfrac{1}{2}N+1,..,3N/2-1\\
\hat{X}(3N/2) &= \tfrac{1}{2}X(N/2)\\
\hat{X}(3N/2+k) &= X(N/2+k) && k=1,..,\tfrac{1}{2}N-1
\end{aligned} \qquad (3.9)$$

Thus we have effectively "pulled" the spectral patches twice as far apart. The only modification of the spectrum is the splitting of the central frequency component $X(N/2)$ to satisfy equation (3.8). Note that $X(0)$ and $X(N/2)$ are always real for real x which follows directly from the definition of the DFT:

$$x(j) = \sum_{k=0}^{N-1} X(k)e^{i2\pi jk/N}$$

$$X(k) = \frac{1}{N}\sum_{j=0}^{N-1} x(j)e^{i2\pi jk/N} \qquad (3.10)$$

Let us now evaluate $\hat{x}$ in terms of x:

$$\hat{x}(p) = \sum_{j=0}^{2N-1} \hat{X}(j)e^{i2\pi pj/2N}$$

$$= \sum_{j=0}^{\frac{1}{2}N-1} X(j)e^{i2\pi jp/2N} + \tfrac{1}{2}X(N/2)\{e^{i2p\frac{N}{2}\frac{1}{2N}\pi} + e^{i2p\frac{3N}{2}\frac{1}{2N}\pi}\}$$

$$+ \sum_{j=N/2+1}^{N-1} X(j)e^{i2\pi jp/2N}e^{i2\pi Np/2N}$$

$$= \sum_{j=0}^{N/2-1} e^{i\pi jp/N}\{\frac{1}{N}\sum_{l=0}^{N-1} x(l)e^{-i2\pi jl/N}\}$$

$$+ e^{i\pi p}\sum_{j=N/2+1}^{N-1} e^{i\pi jp/N}\{\frac{1}{N}\sum_{l=0}^{N-1} x(l)e^{-i2\pi jl/N}\}$$

$$+ \tfrac{1}{2}\{e^{i\pi p/2} + e^{i3\pi p/2}\}\frac{1}{N}\sum_{l=0}^{N-1} x(l)e^{-i2\pi\frac{N}{2}l/N}$$

$$= \frac{1}{N} \sum_{l=0}^{N-1} x(l)\Big[\sum_{j=0}^{N/2-1} e^{i\pi j(p-2l)/N} + (-1)^p \sum_{j=N/2+1}^{N-1} e^{i\pi j(p-2l)/N}$$

$$+ \tfrac{1}{2}(e^{i\pi p/2} + e^{i3\pi p/2})e^{-i\pi l} \Big]$$

Consider p even:

$$\hat{x}(p) = \frac{1}{N} \sum_{l=0}^{N-1} x(l)\Big[\sum_{j=0}^{N-1} e^{i\pi j(p-2l)/N} + \sum_{j=N/2+1}^{N-1} e^{i\pi j(p-2l)/N}$$

$$+ e^{i\pi(p/2-1)\frac{N}{2}\cdot\frac{2}{N}} \Big]$$

since $e^{i\pi p/2} + e^{i3\pi p/2} = e^{i\pi p/2} + e^{-i\pi p/2} = 2e^{i\pi p/2}$ for p even. So

$$\hat{x}(p) = \frac{1}{N} \sum_{l=0}^{N-1} x(l) \sum_{j=0}^{N-1} e^{i\pi j(p-2l)/N}$$

$$= \frac{1}{N} \sum_{l=0}^{N-1} x(l) N\delta_{p,2l} \qquad \text{by orthogonality of } e^{i\pi jk/N}$$

$$= x(p/2)$$

Hence the even points of the interpolated array $\hat{x}$ are identical with the points of the original array x as we desire.

The case of p odd may be similarly determined (these are the interpolated points)

$$\hat{x}(p) = \frac{1}{N} \sum_{l=0}^{N-1} x(l)\Big[\sum_{j=0}^{N/2-1} e^{i\pi j(p-2l)/N} - \sum_{j=N/2+1}^{N-1} e^{i\pi j(p-2l)/N} \Big]$$

since $e^{i\pi p/2} + e^{i3\pi p/2} = 0$ for p odd. On simplifying this reduces to

$$\hat{x}(p) = \frac{1}{N} \sum_{l=0}^{N-1} x(l) \cdot (-1)^{(p-2l-1)/2} \cdot \frac{1+\cos(\pi(p-2l)/N)}{\sin(\pi(p-2l)/N)} \qquad (3.11)$$

This latter result is of little practical interest however. We shall now show how the amount of computation for interpolation may be reduced. We have seen above that the even points in the interpolated array $\hat{x}$ correspond to the points in the original array x. So for q even and $p = q + 1$

$$x(q/2) = \hat{x}(q) = \sum_{j=0}^{2N-1} \hat{X}(j) e^{i2\pi jq/2N} \qquad (3.12)$$

and

$$\hat{x}(p) = \hat{x}(q+1) = \sum_{j=0}^{2N-1} \hat{X}(j) e^{i2\pi (q+1)j/2N}$$

$$= \sum_{j=0}^{2N-1} \{\hat{X}(j) e^{i\pi j/N}\} e^{i2\pi jq/2N} \qquad (3.13)$$

Consider

$$x'(q/2) = \hat{x}'(q) = \sum_{j=0}^{2N-1} \hat{X}'(j) e^{i2\pi qj/2N}$$

where

$$\hat{X}'(j) = \hat{X}(j) e^{i\pi j/N}$$

Hence

$$x'(q/2) = \hat{x}'(q) = \hat{x}(q+1) \qquad (3.14)$$

This is in fact a verification of the Shifting Theorem for the DFT. Thus we see that by multiplying the extended spectrum $\hat{X}(j)$ by $e^{i\pi j/N}$ we shift the odd points to the even points (and *vice versa*) which correspond to the points of x. Hence we may determine the factors by which the original spectrum $X(j)$ should be multiplied to calculate the interpolated points.

Clearly they are

$$\begin{aligned} & e^{i\pi j/N} && j = 0,\ldots,\tfrac{1}{2}N-1 \\ \tfrac{1}{2}(e^{i\pi/2}+e^{-i\pi/2}) &= 0 && j = \tfrac{1}{2}N \\ e^{i\pi(j+N)/N} &= -e^{i\pi j/N} && j = \tfrac{1}{2}N+1,\ldots,N-1 \end{aligned} \qquad (3.15)$$

Hence if it is desired to interpolate an array x, one determines the DFT, X, multiplies by the given factors and inverse transforms. The array will now comprise the interpolated points. This involves an N point DFT ($N\log_2 N$ complex multiplications), N complex multiplications for filtering and an inverse DFT ($N\log_2 N$ complex multiplications), a total of $2N\log_2 N + N$. If the 2N point array is formed then $N\log_2 N + 2N\log_2 2N = 3N\log_2 N + 2N$ complex multiplications are required. Thus the modified method results in a saving of about 30%. However if one desires to form an interpolated array from a filtered spectrum, then both the standard points and the interpolated points will have to be determined. In this case the method requires $N\log_2 N + N + N\log_2 N = 2N\log_2 N + N$ compared to $2N\log_2 2N = 2N\log_2 N + 2N$ complex multiplications, so that the savings are small, and hardly worthwhile unless optimum efficiency is essential.

Let us now consider various techniques for restoring an image whose degradation may be described by equation (3.3). We shall mention methods of processing space-variant degradations at a later stage.

The imaging system transfer function H, the Fourier transform of the impulse response h, generally decreases in amplitude fairly rapidly with frequency so that it is approximately zero at high frequencies. Hence if one attempts to restore an image by dividing

equation (3.4) by H giving

$$\hat{F} = \frac{G}{H} - \frac{N}{H} \qquad (3.16)$$

one may expect that $\hat{F}$ will be a very poor approximation to F, in fact it will probably be far worse than G, since, as was mentioned above, the noise spectrum tends to be flat so that dividing by a function tending to zero will amplify high frequency noise components enormously, completely swamping the image information. In practice then one tries to restore low frequencies accurately while attenuating high frequencies as a compromise between smoothing and restoration. As with optimum smoothing described above, we may determine the optimum linear filter for obtaining a best mean square approximation $\hat{f}$ to f. Comparing (3.4),(3.5) and (3.7) we see that S must be replaced by $F.H$ in (3.7). Hence

$$\Phi_S = |S|^2 = |F|^2|H|^2 = \Phi_F|H|^2$$

Thus the Wiener filter becomes

$$R = \frac{\Phi_F|H|^2}{\Phi_F|H|^2 + \Phi_N}$$

$$= \frac{|H|^2}{|H|^2 + \Phi_N/\Phi_F}$$

This recovers a best estimate of $S = F.H$ while we require $F = S/H$. Hence the required filter is

$$R' = \frac{R}{H} = \frac{H^*}{|H|^2 + \Phi_N/\Phi_F} \qquad (3.17)$$

thus we perform an optimal smoothing and then use the

ideal system inverse H^{-1}, though naturally the operations are performed simultaneously by the filter. Inspection of the Wiener filter shows that in regions where the system transfer function H is small or the noise spectrum is large compared with the signal, the filter tends to zero. Hence it places most weight on those parts of the image spectrum which contain the highest proportion of pure signal spectrum. While many different restoring filters and techniques have been proposed, they are more or less equivalent in effect, as one would expect. Once again computation and convenience will determine which method should be used in a given situation.

An effective iteration technique has been described by Iinuma[15] and Iinuma and Nagai[16] for image restoration of radioisotope imaging systems. The two papers describe equivalent processes, the former using multiplication in the frequency domain and the latter an iterative deconvolution in the space domain, so that the two methods are in fact Fourier transforms of each other. We shall consider the frequency domain approach as it is superior in general (for example one may simultaneously smooth the image or utilize other spectral filters).

The process is based on the following identity

$$\frac{1}{H} = \frac{1}{1-(1-H)} = \sum_{i=0}^{\infty} (1-H)^i \qquad (3.18)$$

Now this expression will converge only for $0<H<2$. For $H \to 0$, the series diverges to infinity as does the closed form. For $H = 2$ the series oscillates. Iinuma makes no mention of this though clearly the method will fail if H does not satisfy these conditions. Since we assume energy conservation in the imaging system, the impulse response must have unit area, so that the DC component of the spectrum is unity. Inspection of

the bounds on H stated above indicates that the method may only be used in cases where H is nonnegative with no part exceeding twice the DC value. Thus it could not be used for restoring defocussed images or linear motion blur for example, where negative values occur. One might note in passing that a nonnegative impulse response always has a spectrum whose upper bound is the DC value since

$$\begin{aligned} |H(f)| &= \left| \int_{-\infty}^{\infty} h(x) e^{-i2\pi x f} dx \right| \\ &\leq \int |h(x) e^{-i2\pi x f}| dx \\ &= \int h(x) dx \quad \text{since } h \geq 0 \\ &= H(0) \end{aligned}$$

Thus usually the only restriction on a transfer function is that it should be nonnegative, which is clearly satisfied by a Gaussian function.

Writing $\hat{F}_j$ as the approximation to F obtained from the j^{th} iteration using (3.18) we have that

$$\hat{F}_j = G \sum_{i=0}^{j} (1-H)^i \qquad (3.19)$$

thus obtaining the recursive formula

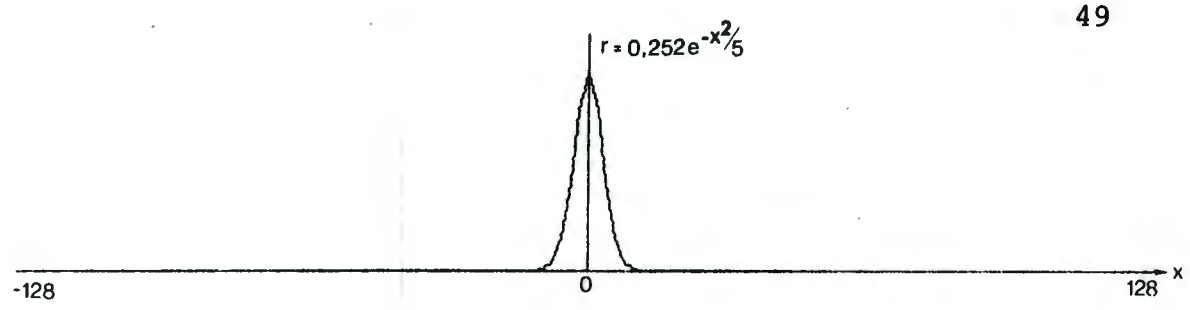
$$\hat{F}_j = G + (1-H)\hat{F}_{j-1} \qquad (3.20)$$

with

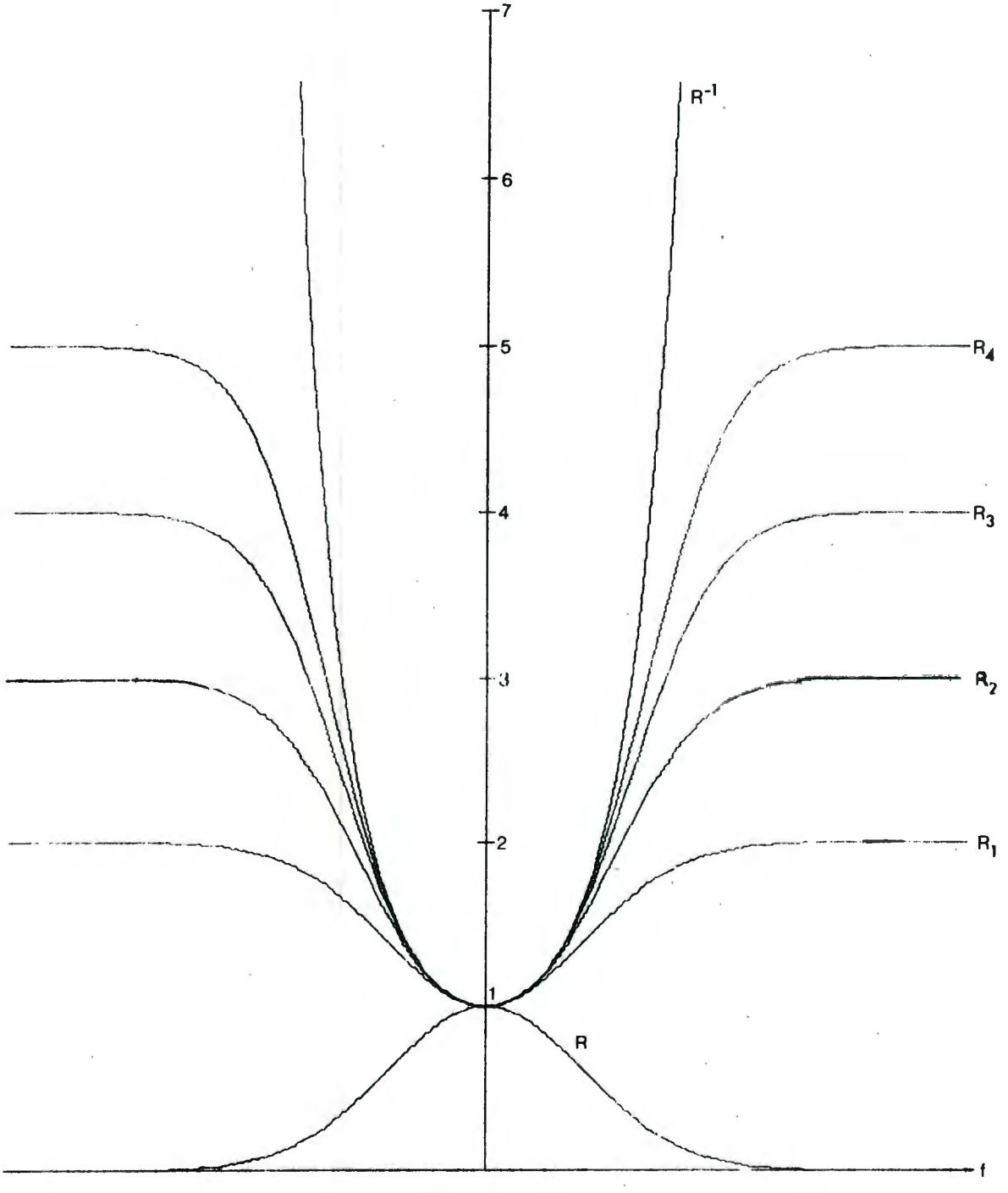
$$\hat{F}_0 = G$$

We may write $F_j = GR_j$ with

$$R_j = \sum_{i=0}^{j} (1-H)^i$$



space domain



frequency domain

Figure 2: Unmodified Iteration Functions

The first four R_i (i=1,..,4) functions are plotted in Fig. 2. The spatial domain Gaussian function is $0{,}252e^{-x^2/5}$ where $128 \leq x \leq 128$. The factor 0,252 normalizes the area under the function to unity to produce a spectrum with $H(0) = 1$. It will be seen that the R_i converge to $R^{-1} = R_\infty = 1/H$.

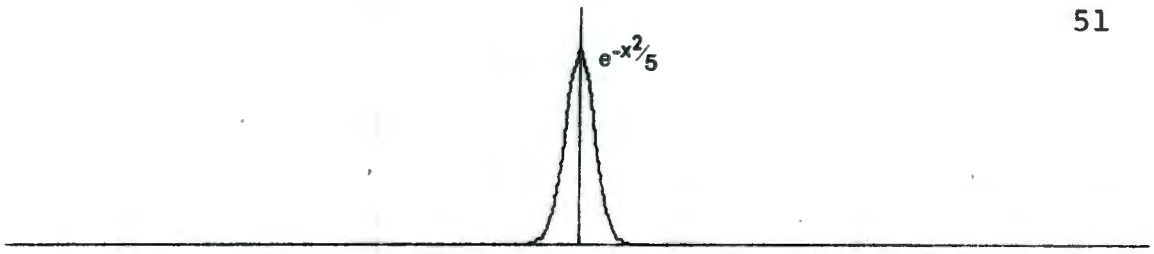
Now if one performs this iterative restoration on an unsmoothed image the resulting images obtained at each stage of the procedure will exhibit increasing noise content as the high frequencies (almost entirely noise) are amplified by the R_i. Thus the image should first be smoothed, a Gaussian function being satisfactory. Iinuma tried several different smoothing functions but made no mention of the fact that performing the restoration on a smoothed image, without modifying the effective system transfer function H, will result in the iteration converging to an incorrect inverse at low frequencies, which seems to be a remarkable oversight. (We know that the iteration will not converge to the inverse at high frequencies - that was the object of smoothing: to reduce amplification of noise). Fig. 3 shows the result of using a smoothing function of $e^{-x^2/1,5}$ with the same impulse response as in Fig. 2, *viz.* $e^{-x^2/5}$. We see that the iteration converges to R'^{-1} rather than the desired R^{-1}, where

$$R'^{-1} = R^{-1}.e^{-1,5\pi^2 f^2} \qquad (3.22)$$

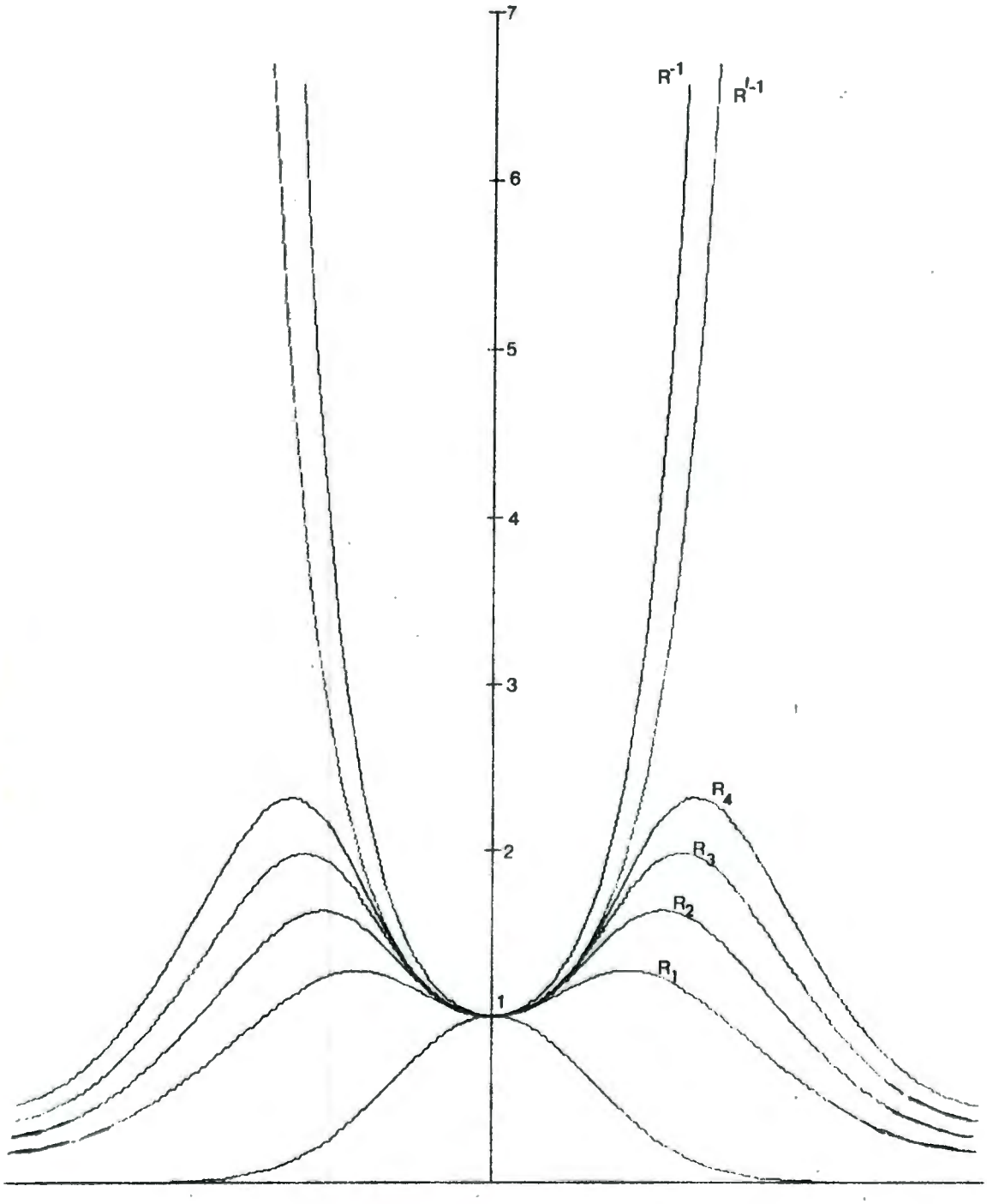
Since the Fourier transform of $e^{-x^2/a}$ is $e^{-a\pi^2 f^2}$, it is clear that, if a smoothing function of $e^{-x^2/b}$ is to be used, the impulse response $e^{-x^2/a}$ should be modified to

$$e^{-x^2/(a+b)} \qquad (3.23)$$

as this transforms to $e^{-(a+b)\pi^2 f^2}$, and the iteration,



space domain



frequency domain

Figure 3: Iteration Functions with Smoothing (Uncompensated)

after smoothing and using this function as H', will converge to

$$e^{(a+b)\pi^2 f^2} e^{-b\pi^2 f^2}$$

$$= e^{a\pi^2 f^2}$$

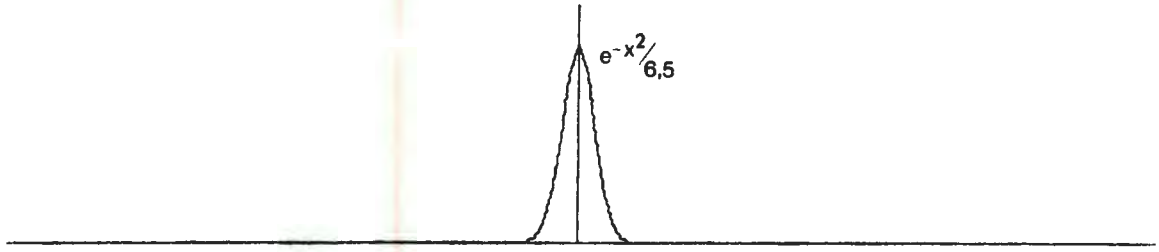
$$= H^{-1}$$

This is illustrated in Fig. 4 where the same smoothing function $e^{-x^2/1,5}$ is used, but the response function is modified to $e^{-x^2/(5+1,5)} = e^{-x^2/6,5}$. Now the iteration converges to R^{-1}. Fig. 5 shows the restoration functions for a smoothing function $e^{-x^2/2,5}$ and a modified response $e^{-x^2/(5+2,5)} = e^{-x^2/7,5}$.

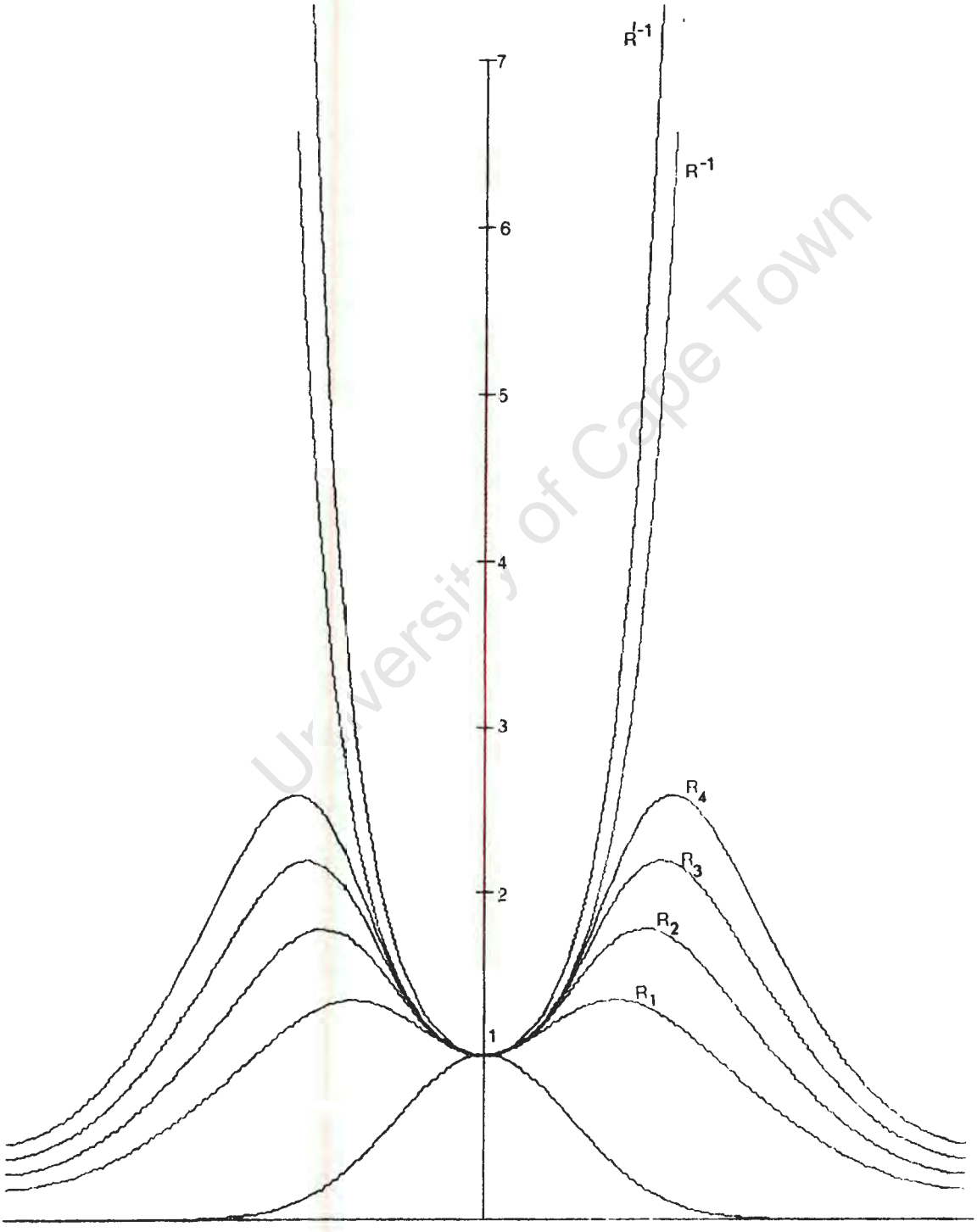
This restoration technique has been used to process images obtained from a Gamma Camera at the Dept. of Nuclear Medicine at Groote Schuur Hospital. The images are obtained as 64×64 digitized arrays which are written onto magnetic tape by their PDP computer. The tape is then read onto disc at the UCT Computer Centre. A program for converting the PDP formatted data to Univac format is given in the Appendix.

To obtain an estimate of the impulse response of the system, a small radioactive source was placed about 8cm from the Gamma Camera and the distribution recorded. The central count distribution is given in Fig. 6. Since the distribution is Gaussian to a good approximation, it was decided to use a Gaussian function as the processing impulse response, and $h(r)$ was empirically determined to be $e^{-r^2/3,5}$. This is sufficiently accurate as tests with different values indicated that the process is insensitive to small variations in the factor of the exponent.

Plates 9 through 23 show a variety of different scintigrams which have been processed by the iterative restoration technique. A smoothing function $e^{-r^2/2}$

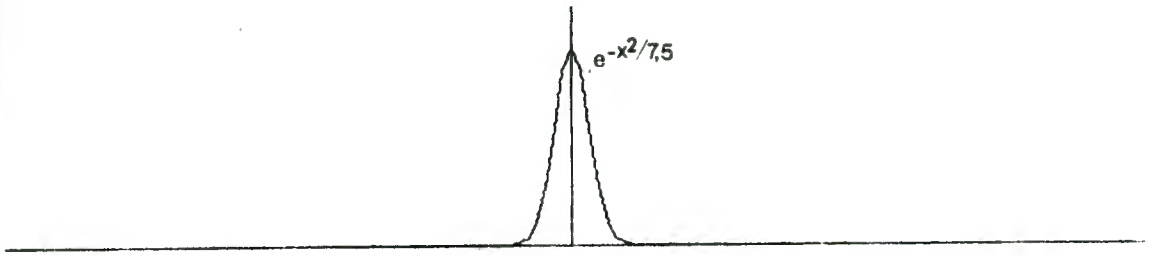


space domain

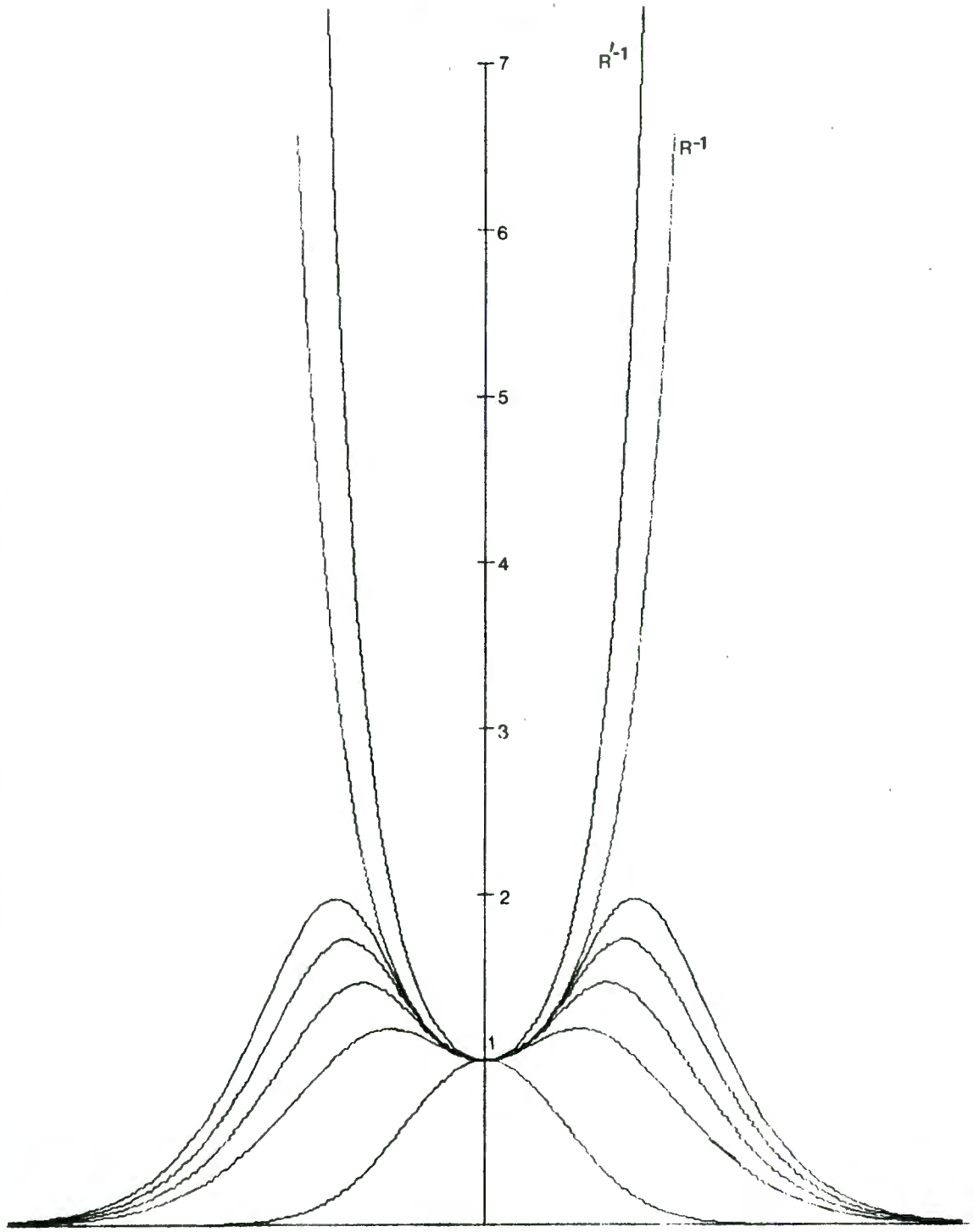


frequency domain

Figure 4: Iteration Functions with Narrow Smoothing (Compensated)



space domain



frequency domain

Figure 5: Iteration Functions with Wide Smoothing (Compensated)

						2	2	5	11	6						
					5	3	3	2	8	1	1					
				1	2	6	6	4	8	4	5	4				
			2	4	5	11	12	14	16	9	8	3	2			
		1	7	3	10	26	30	31	24	14	11	3	1	0		
	3	2	4	8	12	48	71	66	60	35	11	9	5	1	1	
1	4	6	7	9	23	47	122	109	94	53	20	9	6	1	5	5
3	5	3	6	10	38	83	172	231	150	86	32	17	5	1	2	1
4	2	6	5	20	46	75	248	324	221	117	53	18	5	2	2	1
1	4	5	7	15	34	100	191	238	217	89	58	19	7	5	4	5
2	1	6	3	22	20	53	105	149	143	73	40	13	4	8	5	2
	4	4	3	11	18	25	67	84	86	58	24	12	7	3	2	
		1	2	6	9	22	41	39	55	29	14	9	3	0		
			1	0	5	12	13	20	13	20	13	3	2			
				4	6	1	9	6	5	8	10	5				
					4	4	6	2	6	1	6					
						4	5	6	1	6						

Fig 6: Gamma Camera Count Distribution for Point Source.

was used.

The two-colour plots, known as *anaglyphs*, provide a pseudo-three-dimensional image when viewed through the special red and green spectacles provided. This is

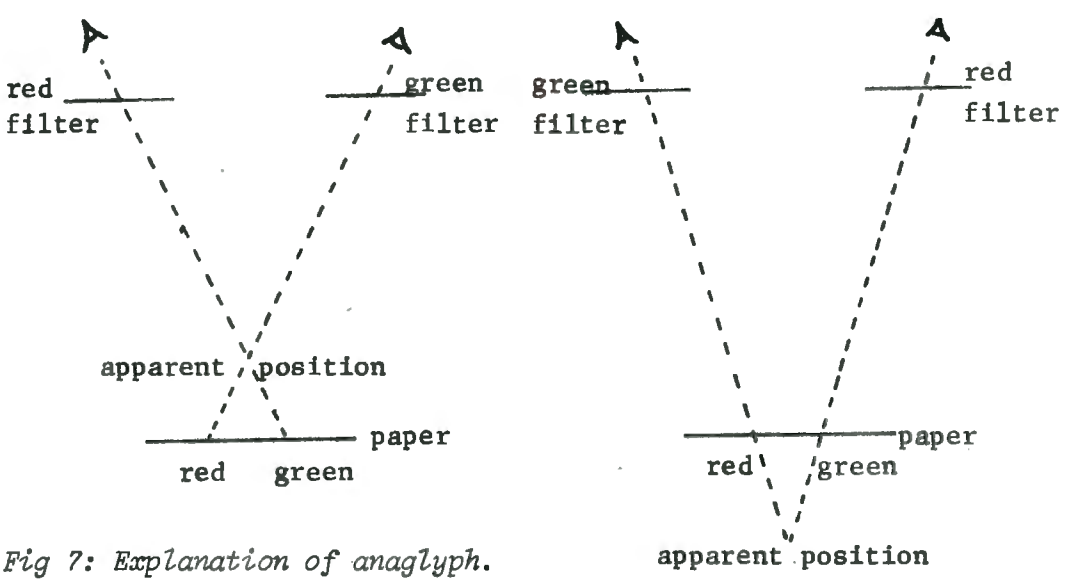


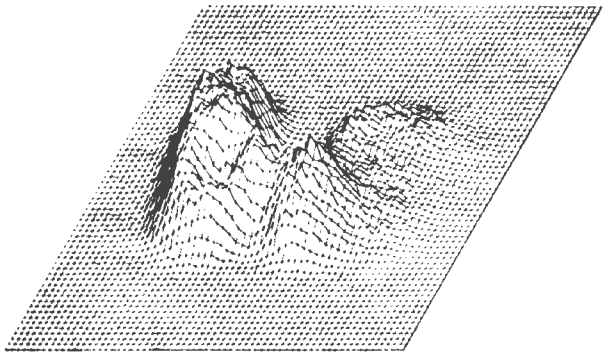
Fig 7: Explanation of anaglyph. Left: correct orientation of spectacles; right: spectacles reversed.

because the brain interprets the displacements of the lines as parallax, so that it assumes that there is one line at the height of intersection of the rays joining the lines to the eyes of the observer, as shown in Fig 7. The greater the line displacement, the higher the resulting image will appear.

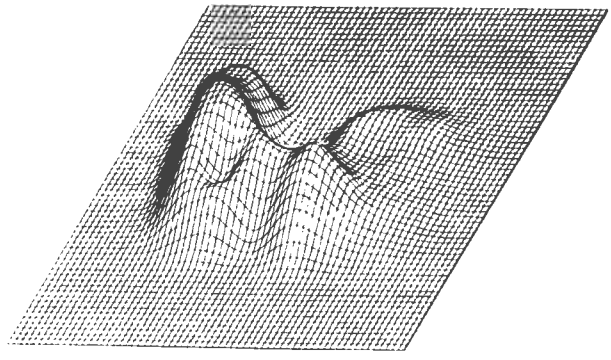
The main point of interest in the restoration pictures is the increase in resolution with the number of iterations. In passing we may mention the "wrap-around" effect associated with the DFT owing to its periodic nature. Inspection of the processed images reveals that this is noticeable in cases where activities on opposite edges of an image differ: after processing the distribution is continuous across the boundary, *i.e.* the lower and higher activities have been increased and decreased respectively. The error extends half the width of the impulse response, which is usually narrow. The effect is seldom troublesome since an image is usually formed with the object of interest in its central region. However if it is necessary to avoid wrap-around, the image may be placed in a larger array comprising the original image with its edge values extended a distance half the width of the impulse response.

The thyroid and lung restorations are most successful. The author prefers the anaglyphs to the isometric displays as all parts of the image are equally visible and they are less susceptible to producing *unwanted* optical illusions (one may note that the three-dimensional image produced by an anaglyph *is* an optical illusion!)

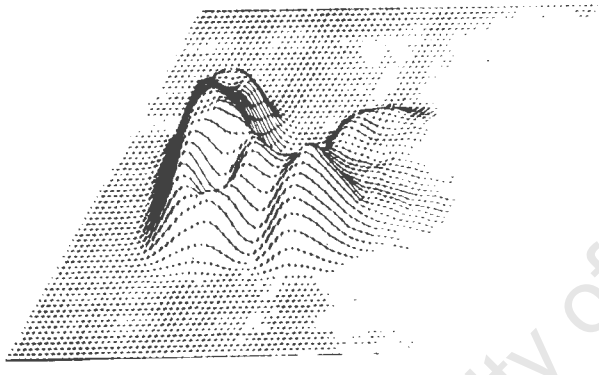
The advantage of using an iteration process in practice is that a sequence of partially restored images is obtained from which one may select the image which displays most clearly the desired information, during the transition from a "blurred" image to those



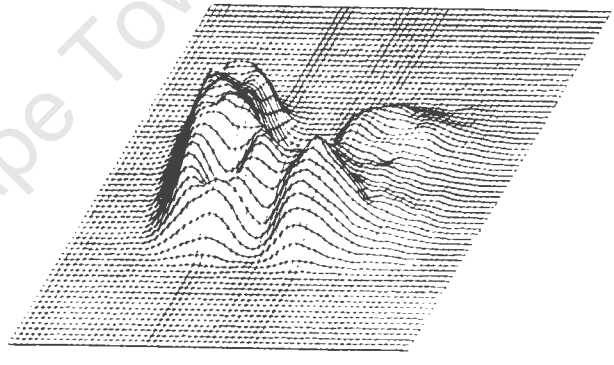
(A) UNPROCESSED PICKER THYROID PHANTOM SCINTIGRAM



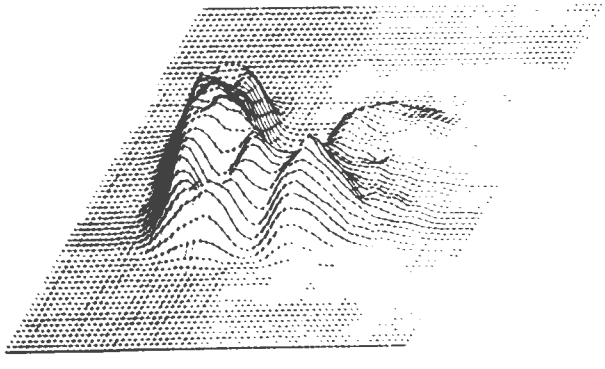
(B) RESULT OF SMOOTHING (A) WITH A NARROW GAUSSIAN FUNCTION (PLATE 3A)



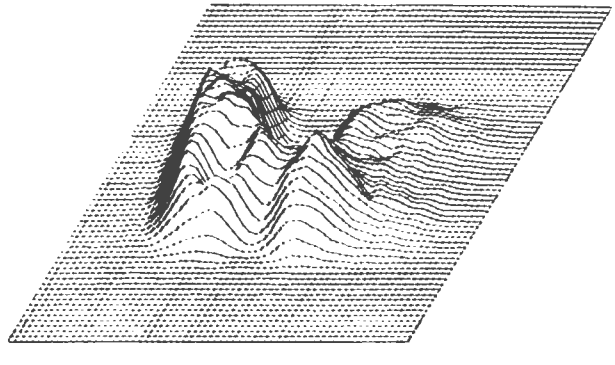
(C) THE IMAGE (B) AFTER ONE ITERATION FOR RESTORATION



(D) THE IMAGE (B) AFTER THREE RESTORING ITERATIONS



(E) AFTER FOUR ITERATIONS



(F) AFTER FIVE ITERATIONS

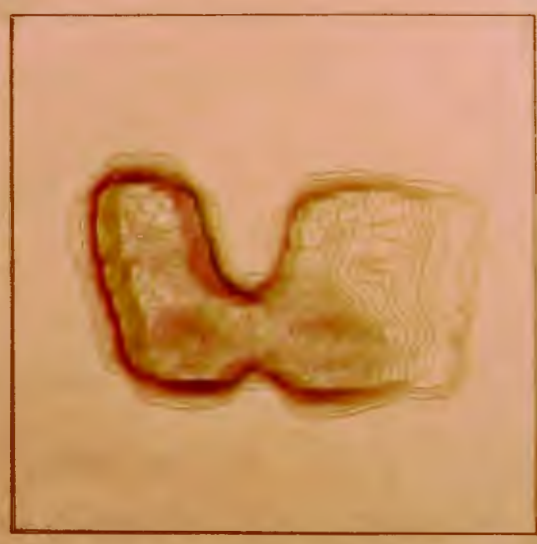
(A) SMOOTHED PICKER PHANTOM (PLATE 9B)

(B) PICKER PHANTOM AFTER 1ST ITERATION (PLATE 9C)

(C) AFTER 3RD ITERATION

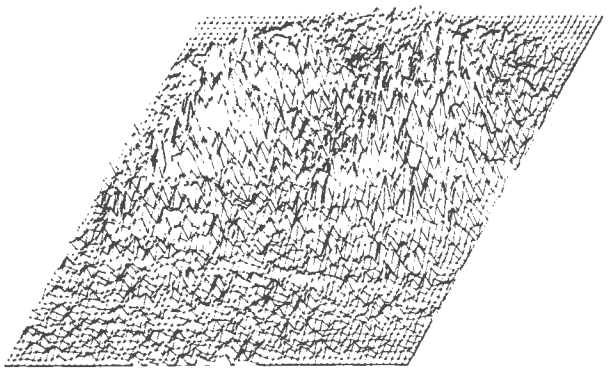
(D) AFTER 4TH ITERATION

(E) AFTER 5TH ITERATION

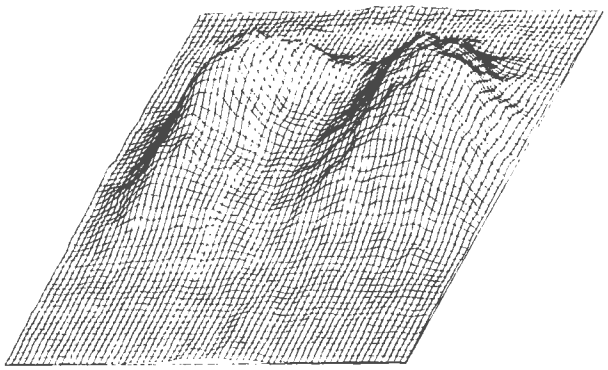


(F) RESTORED PICKER THYROID USING DENSITY SHIFTING

PLATE 10.

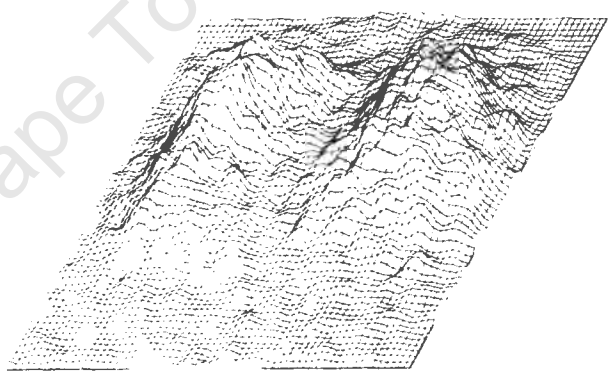


(A) UNPROCESSED P/A LUNG SCINTIGRAM

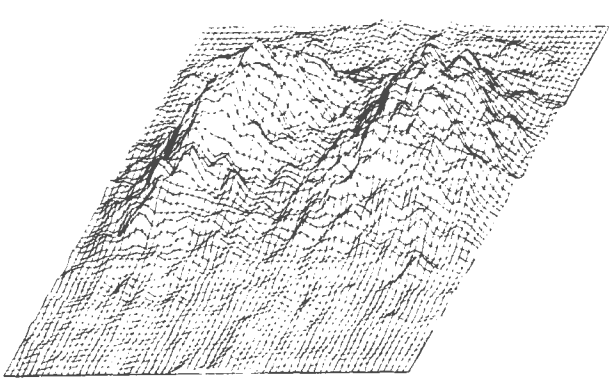


(B) THE RESULT OF SMOOTHING (A)

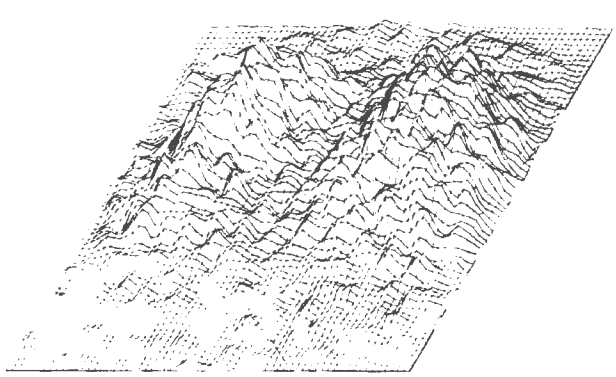
(C) AFTER FIRST RESTORING ITERATION



(D) AFTER SECOND ITERATION



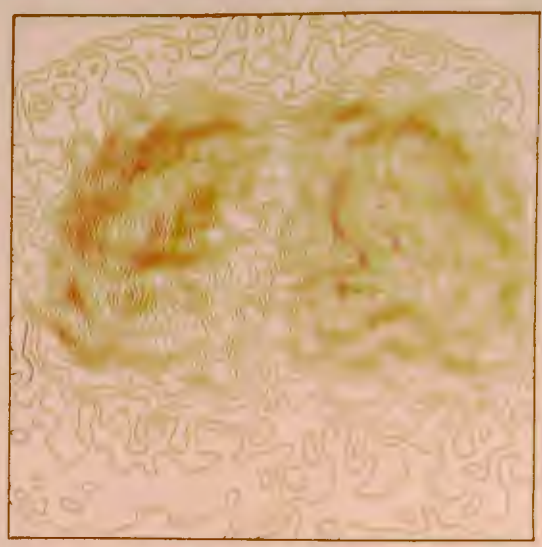
(E) AFTER THIRD ITERATION



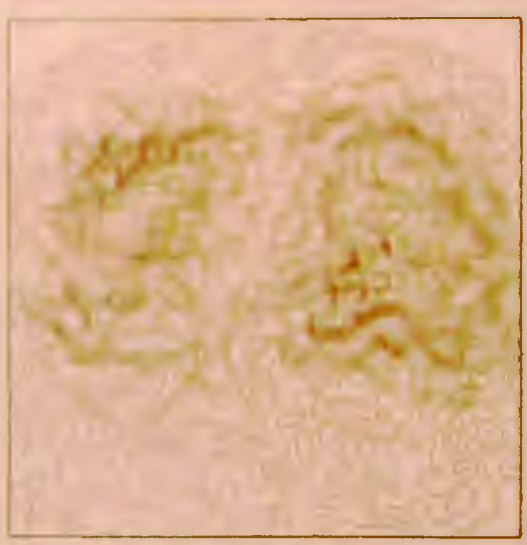
(F) AFTER FIFTH ITERATION

PLATE 11.

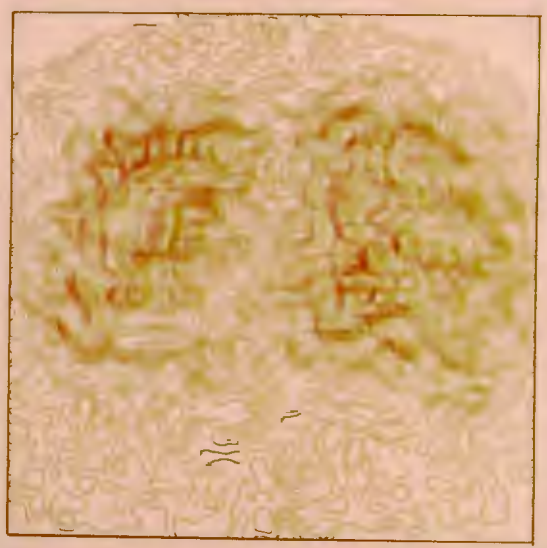
(A) ORIGINAL P/A LUNG PLATE 11B)



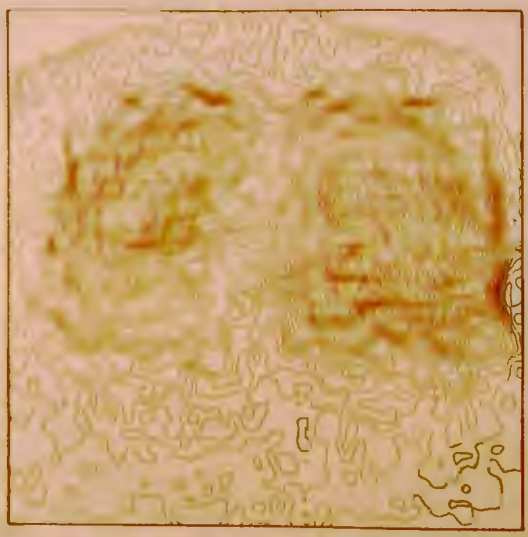
(B) LUNG AFTER 1ST ITERATION



AFTER 3RD ITERATION



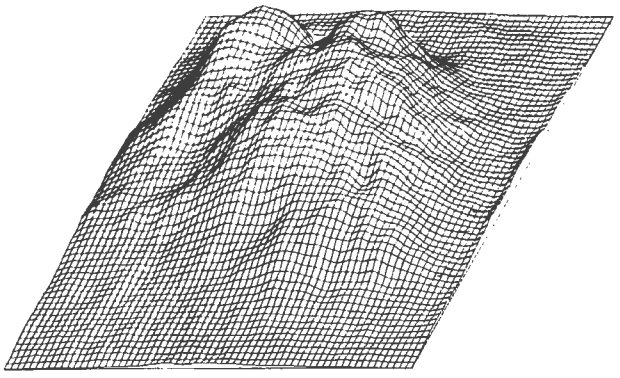
(D) AFTER 5TH ITERATION



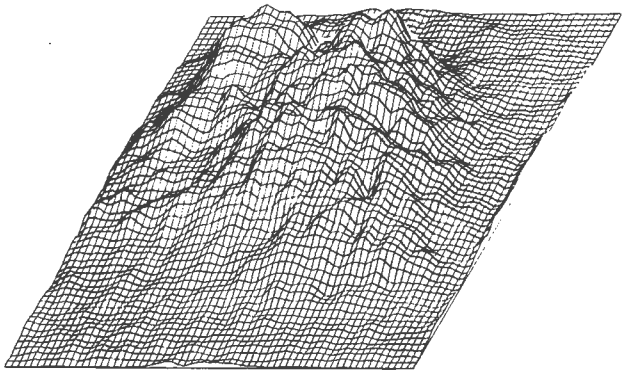
(E) RESTORED P/A LUNG USING OENSITY SHIFTING

(F) AFTER 5TH ITERATION USING MODIFIED RESPONSE $E^{-R^2/5.5}$

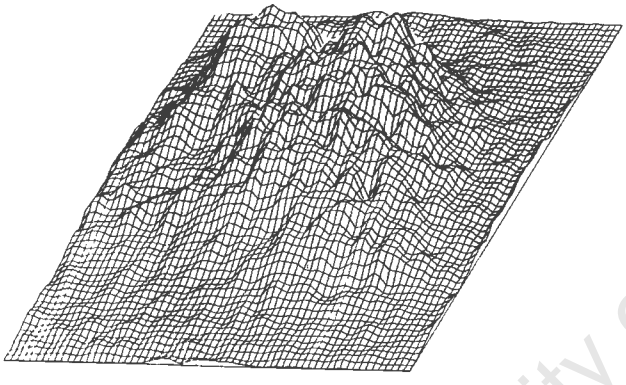
PLATE 12.



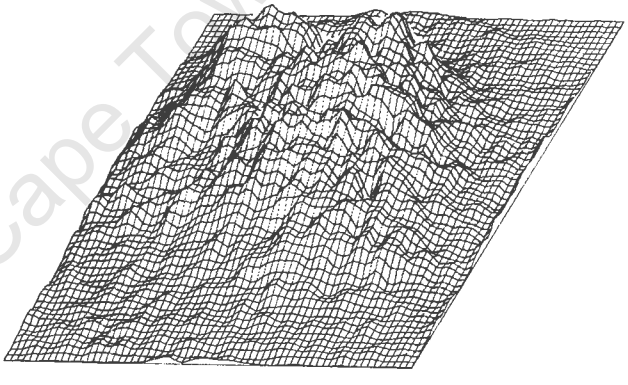
(A) SMOOTHED LEFT OBLIQUE LUNG SCINTIGRAM



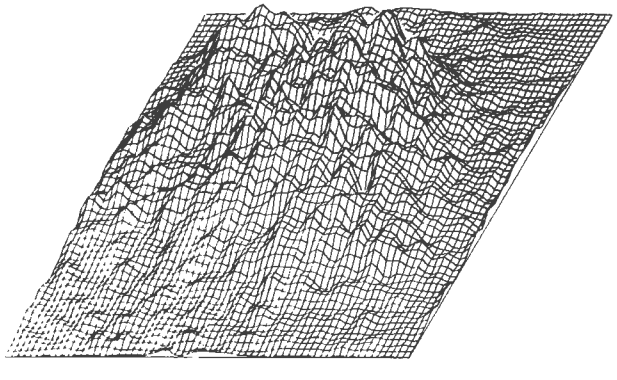
(B) AFTER SECOND RESTORING ITERATION



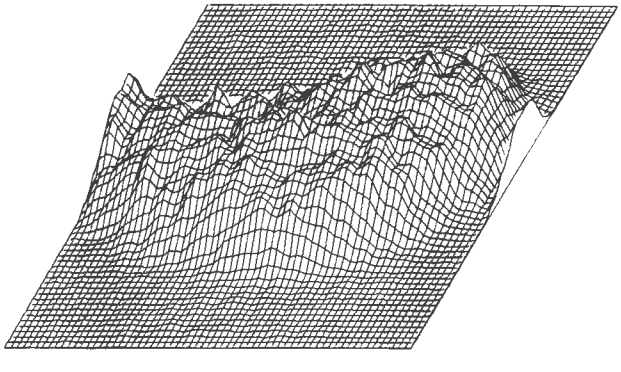
(C) AFTER THIRD ITERATION



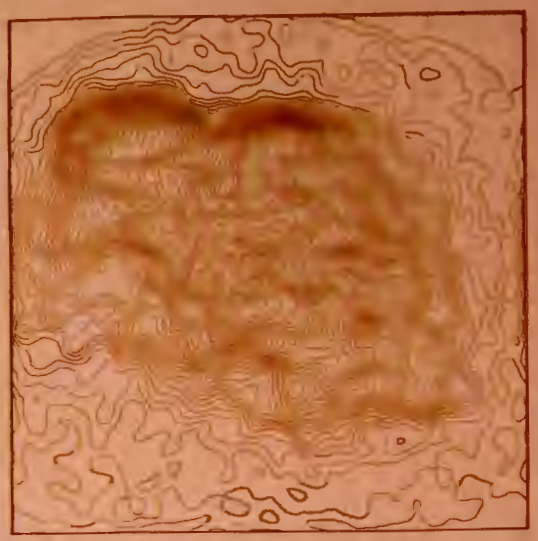
(D) AFTER FOURTH ITERATION



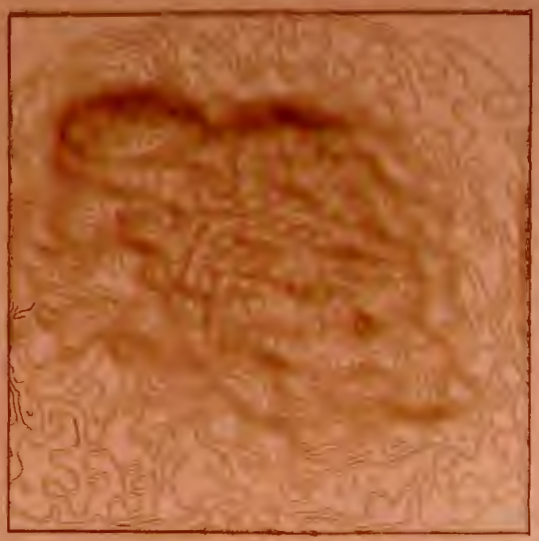
(E) AFTER FIFTH ITERATION



(F) RESTORED LIVER PHANTOM (5 ITERATIONS)

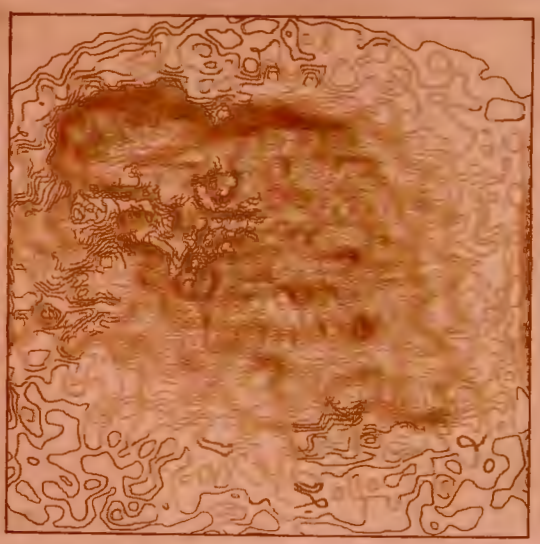


(A) LEFT OBLIQUE LUNG AFTER FIRST RESTORING ITERATION



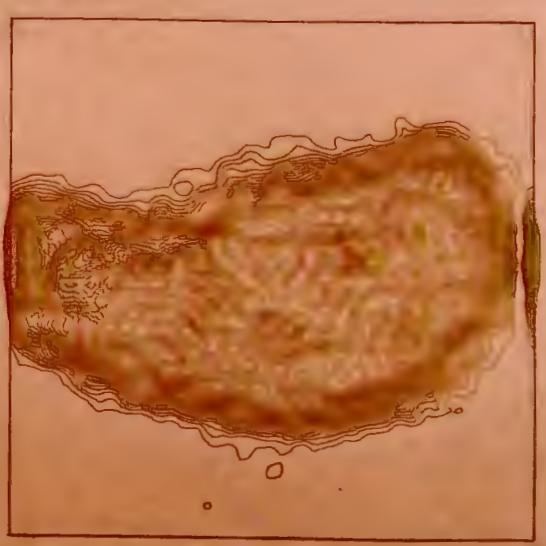
(B) AFTER SECOND ITERATION

(C) AFTER THIRD ITERATION



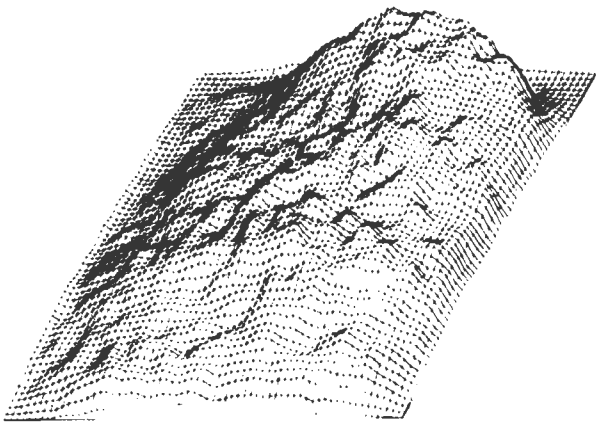
(D) AFTER FOURTH ITERATION

(E) AFTER FIFTH ITERATION

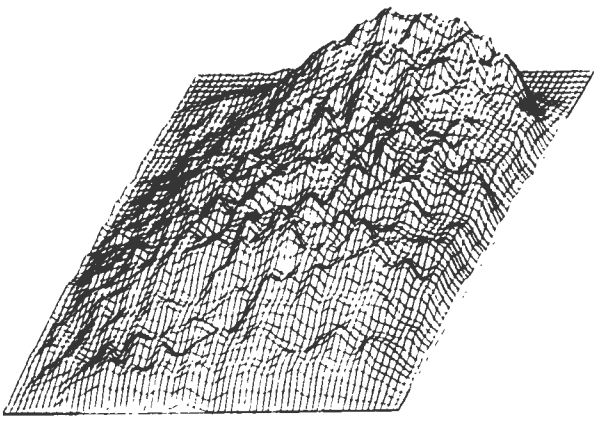


(F) RESTORED LIVER PHANTOM (FIVE ITERATIONS)

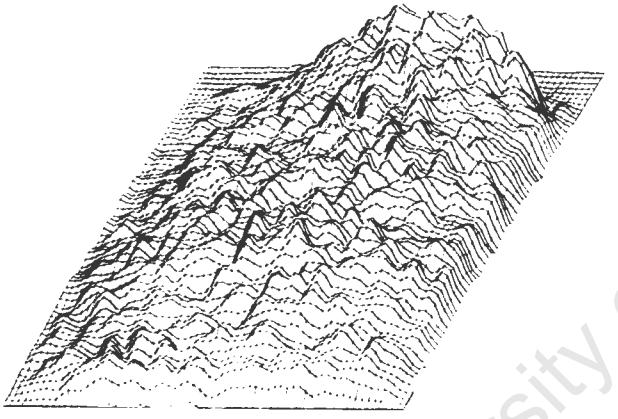
PLATE 14.



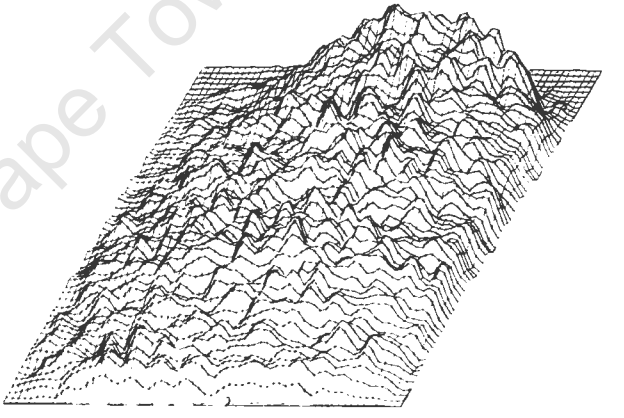
(A) SMOOTHED A/P PELVIS SCINTIGRAM



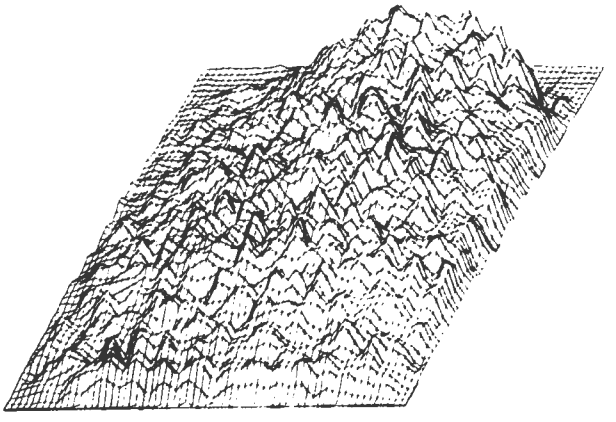
(B) AFTER FIRST ITERATION



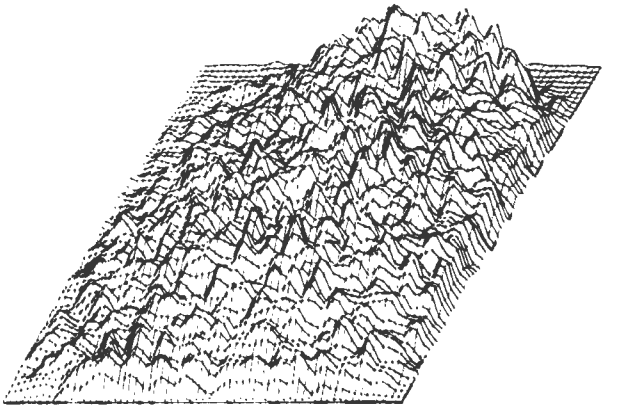
(C) AFTER SECOND ITERATION



(D) AFTER THIRD ITERATION



(E) AFTER FOURTH ITERATION



(F) AFTER FIFTH ITERATION

(A) RESTORED A/P PELVIS AFTER ONE ITERATION

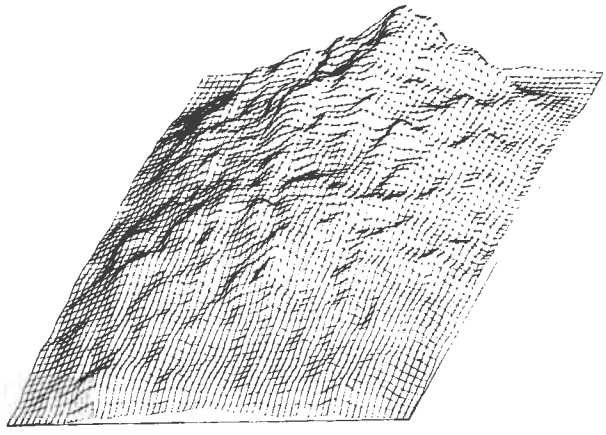
(B) AFTER SECOND ITERATION

(C) AFTER THIRD ITERATION

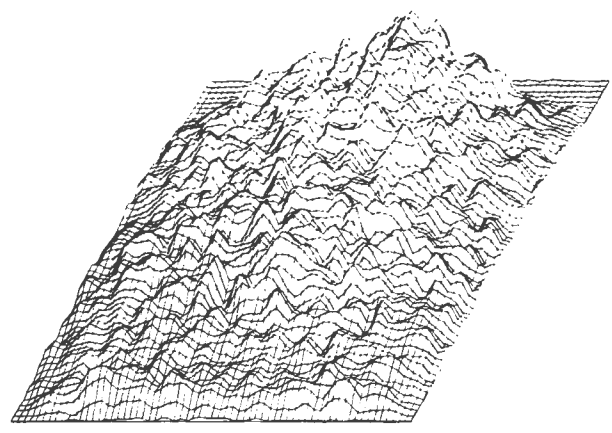
(D) AFTER FOURTH ITERATION

(E) AFTER FIFTH ITERATION

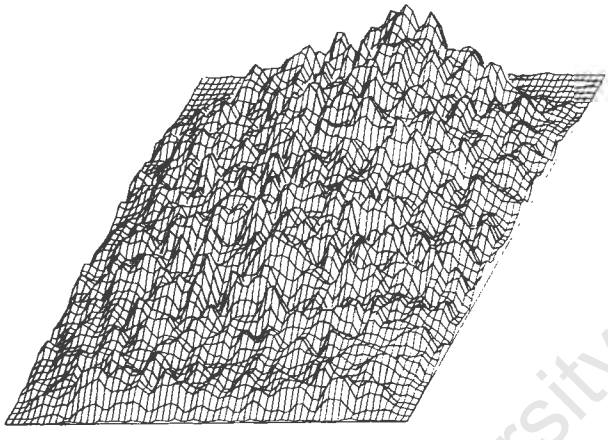
PLATE 16.



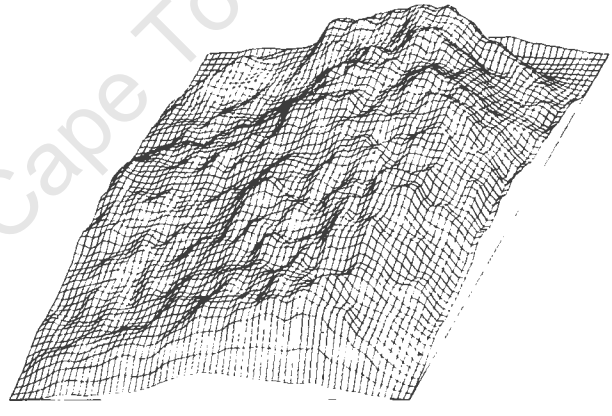
(A) SMOOTHED P/A PELVIS SCINTIGRAM



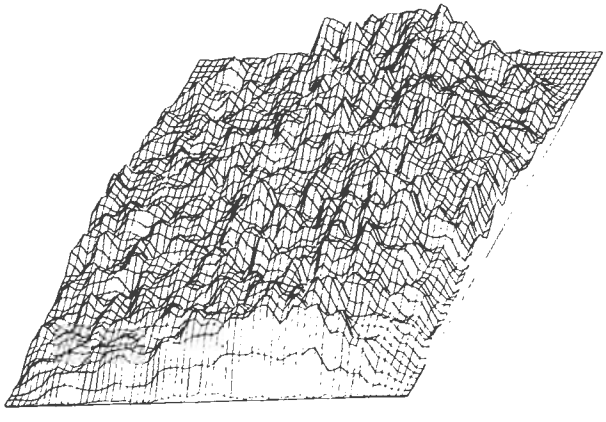
(B) P/A PELVIS AFTER THREE ITERATIONS



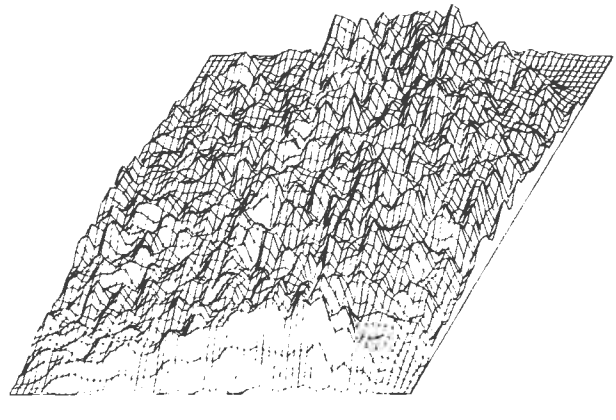
(C) P/A PELVIS AFTER FIVE ITERATIONS



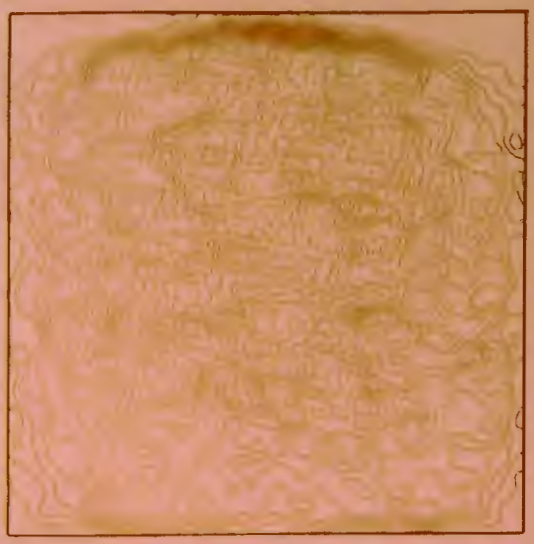
(D) SMOOTHED LATERAL CHEST SCINTIGRAM



(E) LATERAL CHEST AFTER 3 ITERATIONS

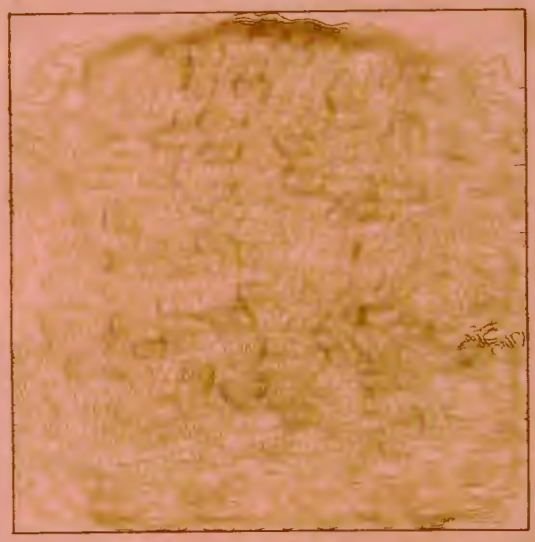


(F) LATERAL CHEST AFTER 5 ITERATIONS

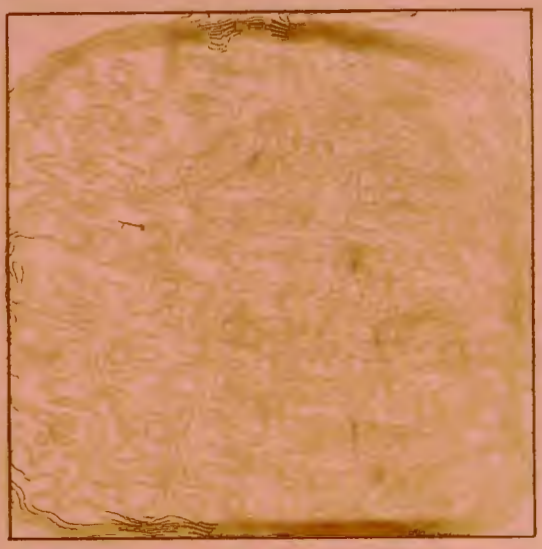


(A) RESTORED P/A PELVIS AFTER 1 ITERATION (PLATE 17)

(B) P/A PELVIS AFTER 3 ITERATIONS



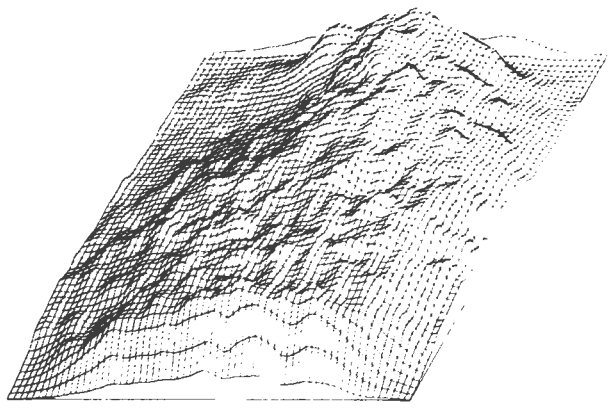
(C) P/A PELVIS AFTER 5 ITERATIONS



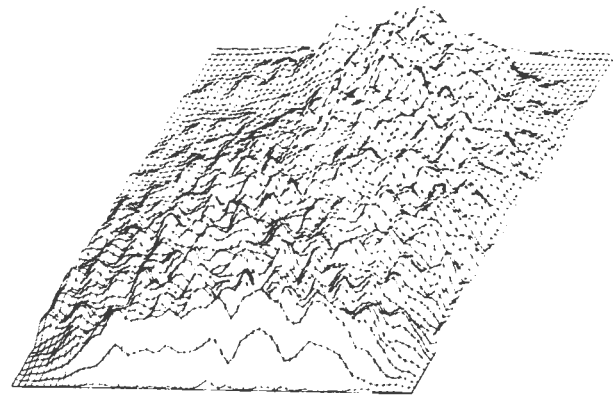
(D) RESTORED LATERAL CHEST AFTER 1 ITERATION (PLATE 17)

(E) LATERAL CHEST AFTER 3 ITERATIONS

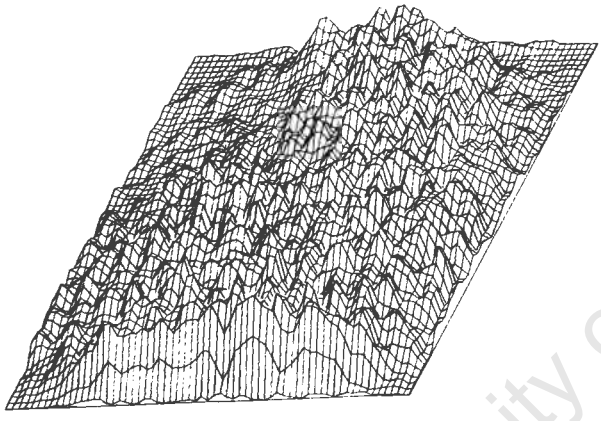
(F) LATERAL CHEST AFTER 5 ITERATIONS



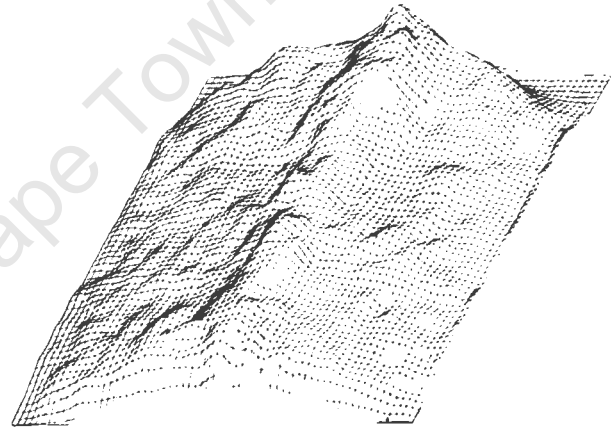
(A) SMOOTHED LATERAL CHEST SCINTIGRAM



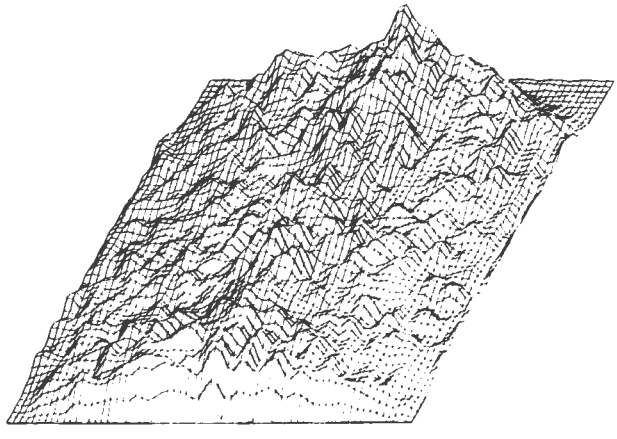
(B) LATERAL CHEST AFTER 3 ITERATIONS



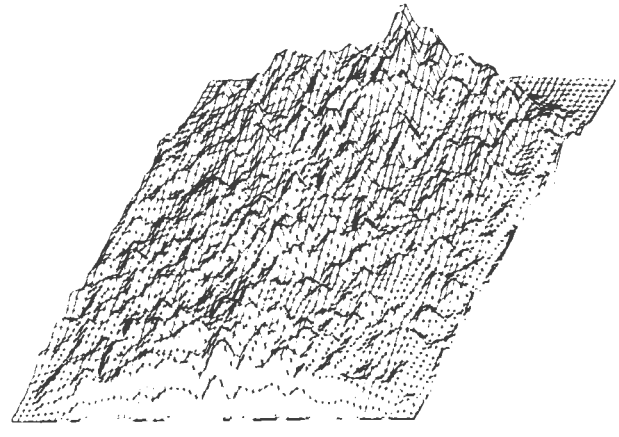
(C) LATERAL CHEST AFTER 5 ITERATIONS



(D) P/A SPINE SCINTIGRAM AFTER SMOOTHING

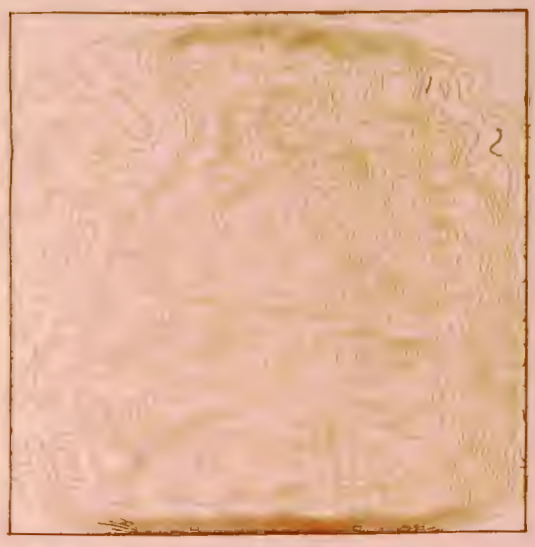


(E) P/A SPINE AFTER 3 ITERATIONS

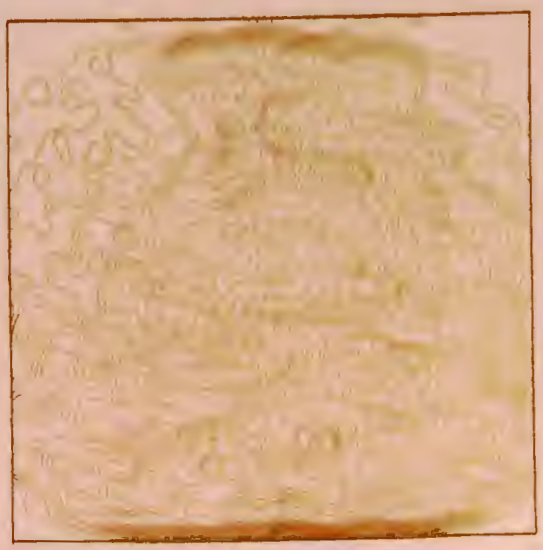


(F) P/A SPINE AFTER 5 ITERATIONS

PLATE 10.



(A) LATERAL CHEST AFTER 1 ITERATION (PLATE 19)



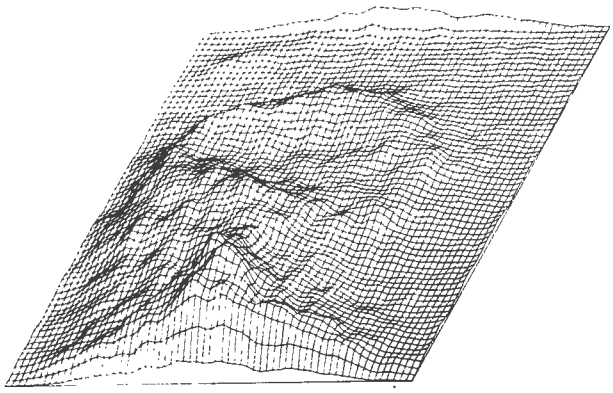
(B) AFTER 2 ITERATIONS

(C) AFTER 3 ITERATIONS

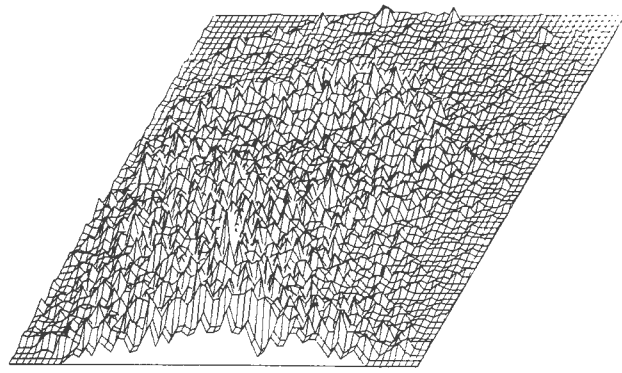
(D) AFTER 4 ITERATIONS

(E) AFTER 5 ITERATIONS

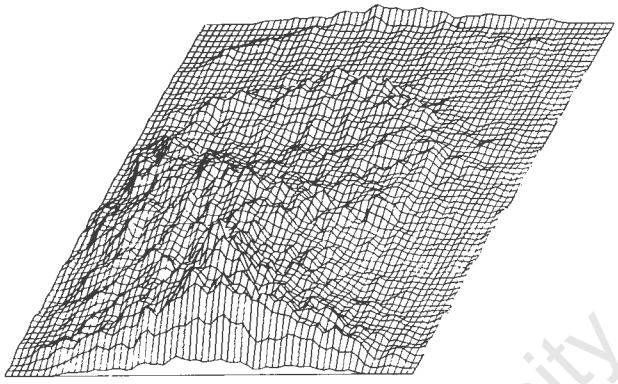
PLATE 20.



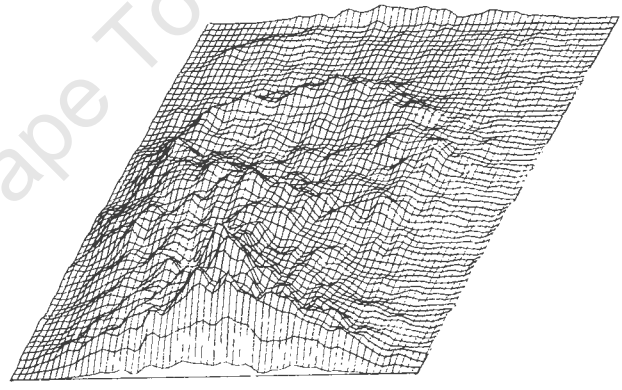
(A) SMOOTHED SKULL



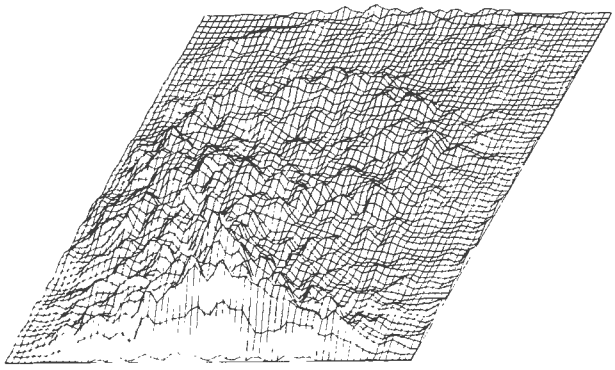
(B) DENSITY SHIFTED SKULL WITHOUT SMOOTHING



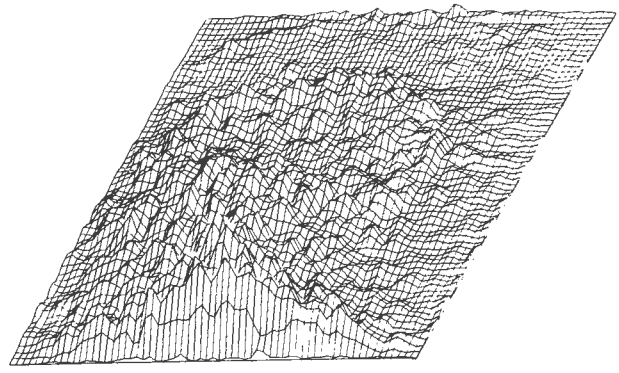
(C) DENSITY SHIFTED SKULL AFTER SMOOTHING



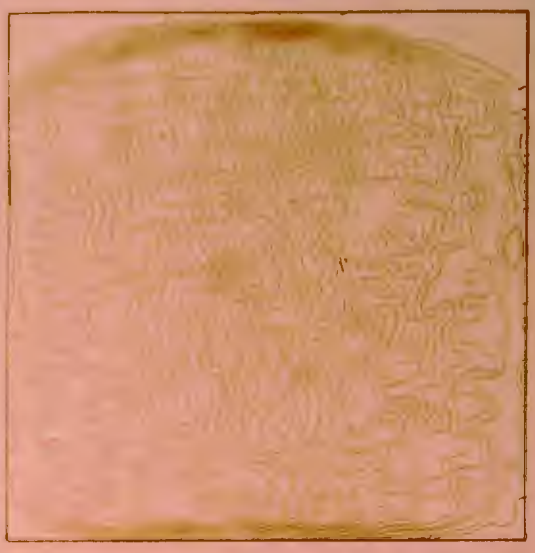
(D) SKULL AFTER 1 ITERATION



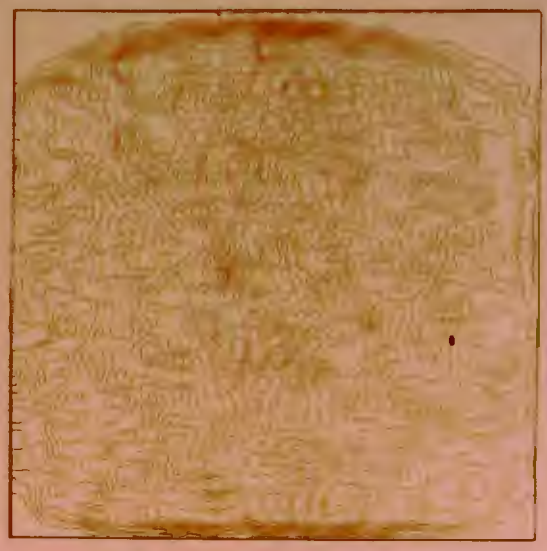
(E) SKULL AFTER 3 ITERATIONS



(F) AFTER 5 ITERATIONS

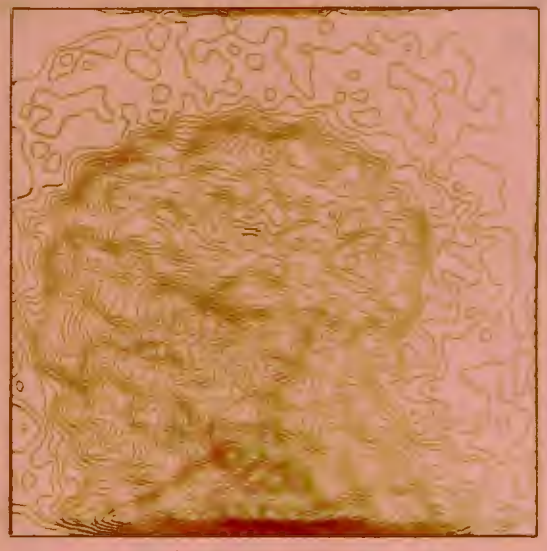


(A) P/A SPINE AFTER 1 ITERATION (PLATE 19)



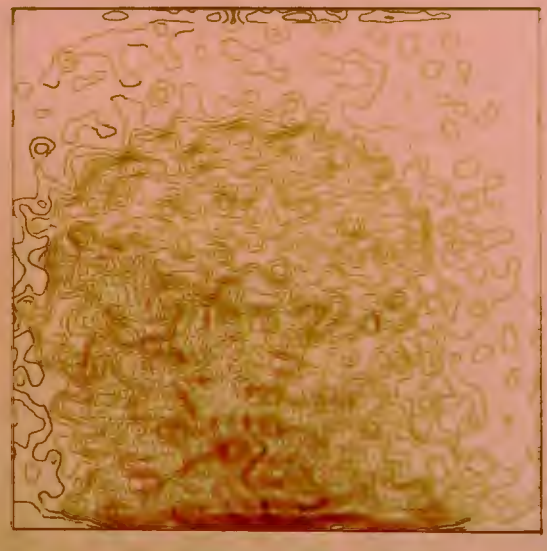
(B) P/A SPINE AFTER 3 ITERATIONS

(C) P/A SPINE AFTER 5 ITERATIONS

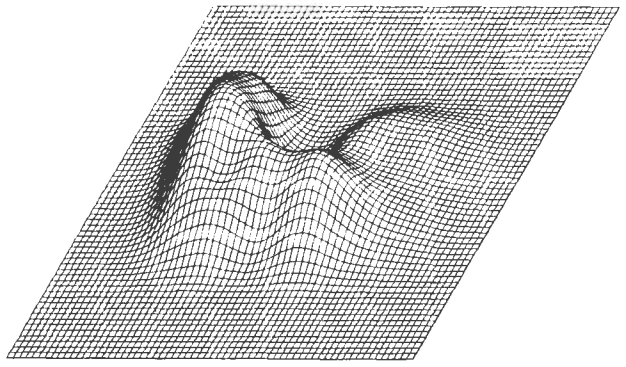


(D) LATERAL SKULL AFTER 1 ITERATION (PLATE 21)

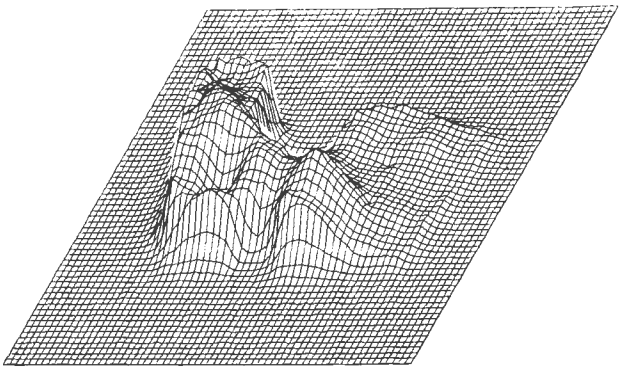
(E) LATERAL SKULL AFTER 3 ITERATIONS



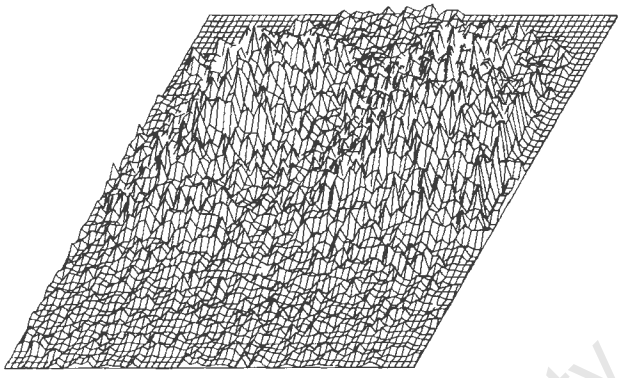
(F) LATERAL SKULL AFTER 5 ITERATIONS



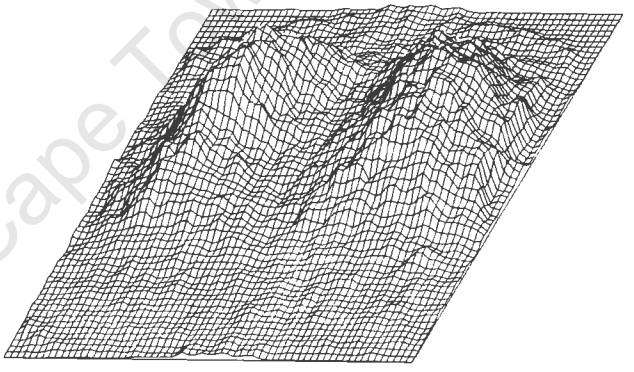
(A) PICKER PHANTOM AFTER SMOOTHING WITH PLATE 3C



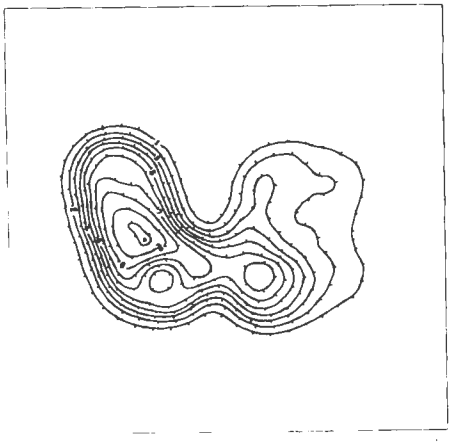
(B) PICKER PHANTOM RESTORED BY DENSITY SHIFTING



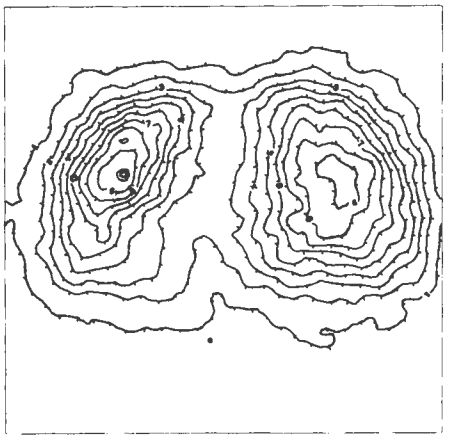
(C) DENSITY SHIFTED LUNG WITHOUT SMOOTHING



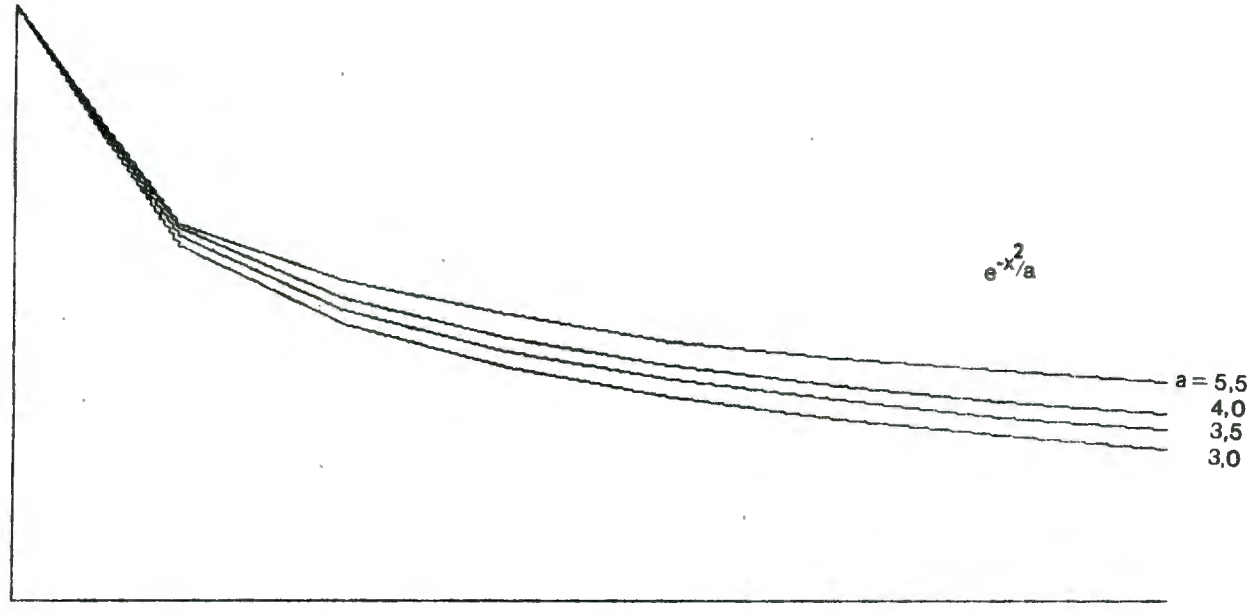
(D) DENSITY SHIFTED LUNG AFTER SMOOTHING



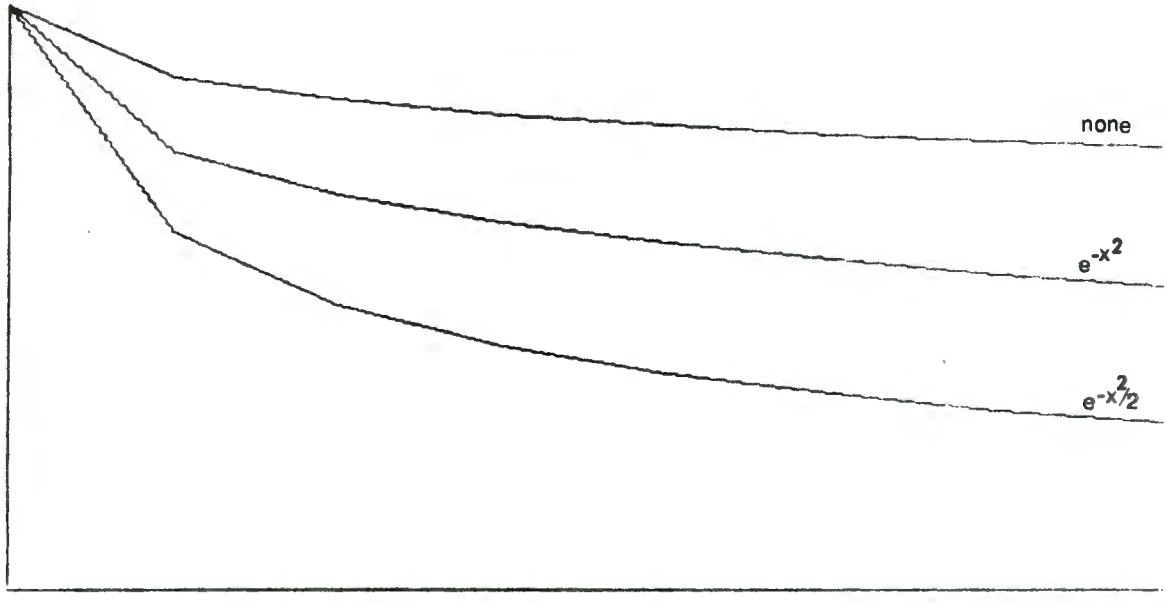
(E) 10-LEVEL CONTOUR PLOT OF PICKER PHANTOM (PLATE 9B)



(F) 10-LEVEL CONTOUR PLOT OF LUNG (PLATE 11B)



Effect of varying impulse response width (constant smoothing)



Effect of varying degree of smoothing (constant impulse response)

Figure 8: Convergence of Iterative Restoration on P/A Lung Scintigram

which are excessively noisy. The speed with which noise becomes too dominant will depend on how smooth the initial image is (after smoothing). However the process is costly owing to the large amount of computation required. Thus one should have a criterion for determining when to terminate the process. Iinuma proposed two means of monitoring the process which give a measure of the difference between successive iteration images; in other words they indicate how much an image differs from the hypothetical image to which the sequence is converging. When the difference is small enough, the process may be automatically terminated. The two inequalities, which in practice give very similar results as would be expected, are

$$\sum^{N^2} (\hat{f}_{n+1} - \hat{f}_n)^2 \leq \sum^{N^2} \hat{f}_0 \qquad (3.24)$$

and

$$\sum^{N^2} (\hat{f}_{n+1} - \hat{f}_n)^2 / \hat{f}_0 \leq N^2 \qquad (3.25)$$

In processing the images shown, (3.25) was modified to

$$\sum^{N^2} (\hat{f}_{n+1} - \hat{f}_n) / \sqrt{\hat{f}_0} \leq N^2 \qquad (3.26)$$

since this represents the mean difference in standard deviations which is appropriate for scintigrams since radioactive sources have a Poisson distribution with the result that noise fluctuations are likely to have an amplitude of the order of a standard deviation (the square root of the activity count at a point) so that variations considerably less than a standard deviation are not significant. Fig. 8 shows the rate of convergence (*i.e.* values assumed by (3.26), normalized to unity for the first iteration) for different smoothing functions and impulse responses when processing the lung of Pl. lla. The lower graphs show the effect of

no smoothing, a smoothing function of e^{-x^2} and a smoothing function $e^{-x^2/2}$ respectively, all for an impulse response of $e^{-x^2/3,5}$. The convergence is very slow without smoothing, and the resulting images are very noisy. The upper graphs show the result of varying the impulse response, while using a smoothing function of $e^{-x^2/2}$. As one would expect, the convergence is fastest for the narrowest response *viz.* $e^{-x^2/3}$. However, this conceals the fact that the most accurate restoration will be provided by the impulse response of $e^{-x^2/5,5}$ for reasons given above during the explanation of the modification required to compensate the impulse response for smoothing.

One objection to the iterative technique is that it uses more computer time by producing a sequence of images rather than one. However one can see that the convergence is rapid for the first 2 or 3 iterations. Furthermore, inspection of the processed images indicates that by the fourth or fifth iteration, the iteration has more or less converged so that there is little if anything to be gained by further iteration. Thus one could in fact either determine the fifth image directly, using R_5 or possibly use R_3 and then iterate normally to find $\hat{f}_3$, $\hat{f}_4$ and $\hat{f}_5$. Clearly then the process can be as economical as any other. Further reductions in computation may be accomplished by modifying the FFT subroutine to utilize the fact that the spectrum of a real function satisfies equation (3.8), so that only half the spectrum need be calculated. In addition the spectrum is modified by a real symmetric filter so that once again only half the spectrum need be filtered.

Another restoration technique is known as density shifting and is a spatial domain convolution[17]. The advantage of a convolutional process is that the impulse response may be space-variant, or may be

varied as a function of the image intensity for example. The principle behind the process is that during image formation the intensity distribution is "smeared out" so to form an approximate inversion, the most likely origin of the activity at each point in the image is determined, and the activity shifted to the probable source.

This may be expressed as

$$(x',y') = \frac{\iint (u,v) f(u,v) h(x-u,y-v)\,du\,dv}{\iint f(u,v) h(x-u,y-v)\,du\,dv} \qquad (3.27)$$

where (x',y') is the point to which the activity at (x,y) is to be moved; *i.e.* (x',y') is the most likely origin of the activity at (x,y); alternatively (x',y') is the centroid of the image weighted by the impulse response centred on (x,y).

The lung, Picker thyroid phantom and one of the two skull scintigrams have been processed by this method using a fixed impulse response $e^{-x^2/3,5}$. After density shifting the images are smoothed with e^{-x^2}. (See Pl. 10f, 12e, 21b&c, 23b,c&d, and 26).

In practice one uses the discrete equivalent of (3.27):

$$(i',j') = \frac{\sum_{k=0}^{N}\sum_{l=0}^{N} (k,l) f(k,l) h(i-k,j-l)}{\sum_{k=0}^{N}\sum_{l=0}^{N} f(k,l) h(i-k,j-l)} + (\tfrac{1}{2},\tfrac{1}{2}) \qquad (3.28)$$

The expression on the right is evaluated and truncated to produce integer values (i',j'). The term $(\frac{1}{2},\frac{1}{2})$ is added to accomplish rounding rather than truncation.

Pizer proposes a method of variable filtering where to reduce the amount of computation the impulse response is approximated by a positive disc surrounded by a negative annulus of equal area. The radius is variable and

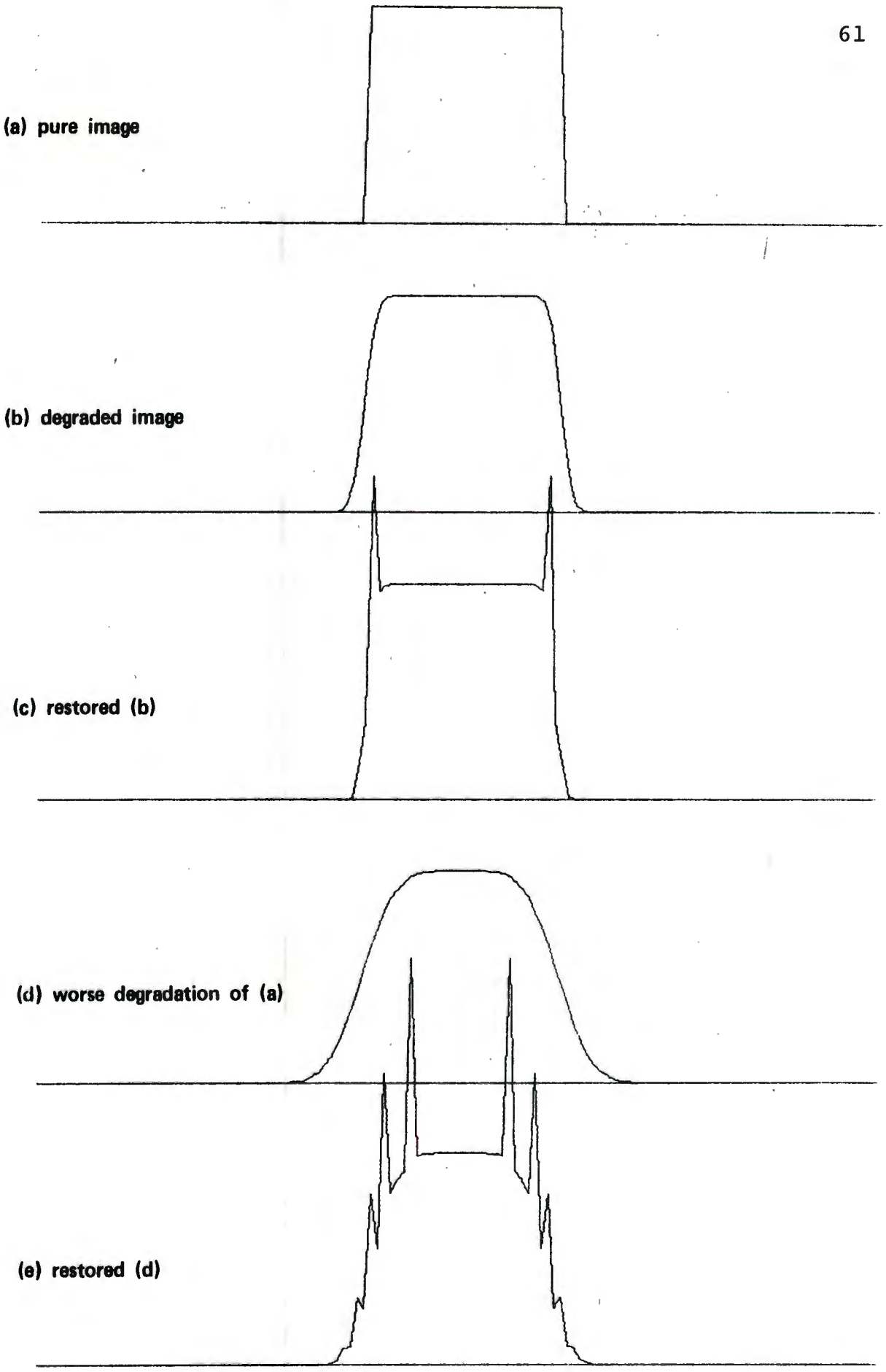


Figure 9: Restoration by Density Shifting

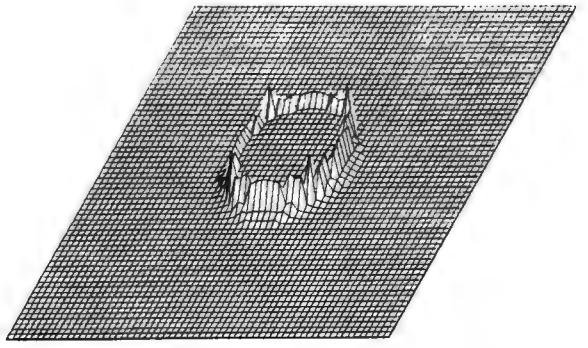
depends on the count density in the neighbourhood. A formula for determining the smallest radius to achieve a given signal to noise ratio is given, as well as the weighting factors for the disc and annulus. The disadvantages of density shifting are that it cannot provide an exact restoration even in the absence of noise and it is designed specifically for a Gaussian impulse response, thus limiting its scope. Pizer claims that the method effectively recovers resolution without producing artifacts. This is doubtful however as illustrated by Fig. 9 and Pl. 24a&b for two-dimensional filtering. Fig. 9 shows two degraded distributions of a rectangular box with their restorations achieved by density shifting. In Pl. 24a irregularities are present as in the one-dimensional case, though they are not so pronounced. Overshoot is still evident after smoothing. The spikes and overshoot are unacceptable. Inspection of the density shifted Picker thyroid phantom images (Pl. 10f and 23b) indicates similar overshoot and anomalous effects on steep slopes which are also apparent in the right side of the density shifted lung image (Pl. 12e). The density shifted images have to be "smeared out" to remove artifacts such as these, thus reducing resolution again. Thus its reliability and accuracy in general are dubious, though the restored images shown here using density shifting appear to be of good quality.

The filters above have all involved point-by-point operations, *e.g.* equation (3.4) may be written as N^2 equations

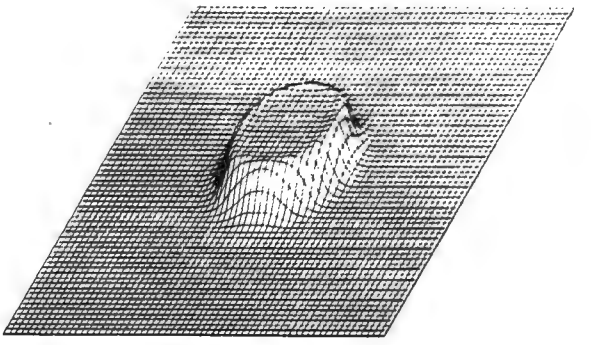
$$G_{ij} = F_{ij} H_{ij} + N_{ij} \qquad i,j=0,1,\ldots,N-1$$

so that these symbols represent *arrays* rather than *matrices*. However the techniques above have matrix equivalents. Expressing equation (3.1) in matrix form we have

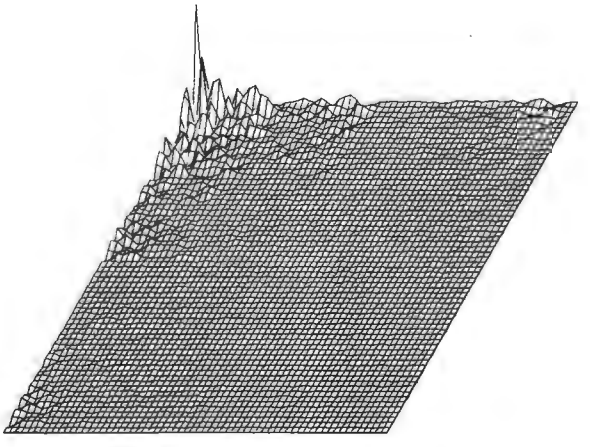
$$g = Hf + n \qquad (3.29)$$



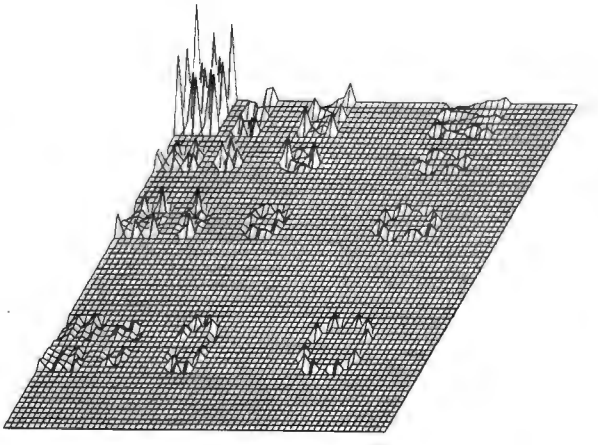
(A) RESTORED DEGRADED PILLBOX BY DENSITY SHIFTING



(B) RESULT OF SMOOTHING (A)



(C) HADAMARD TRANSFORM OF PICKER THYROID



(D) HAAR TRANSFORM OF PILLBOX

PLATE 24

where g, f, n are $N^2 \times 1$ column vectors and H is an $N^2 \times N^2$ matrix. Then the equivalent matrix form of the iterative restoration technique given above is

$$\hat{f}_i = \sum_{j=0}^{i} (I-H)^j g \qquad (3.30)$$

where I is the identity matrix. This is a space domain operation; consequently H may be space-variant. If H is space-invariant then it is block circulant and may be diagonalized by the DFT[18], allowing transform domain computation with its saving in calculation.

Another restoration technique is called constrained least squares filtering. The frequency domain filter is given by

$$R = \frac{H^*}{|H|^2 + \gamma |C|^2} \qquad (3.31)$$

or in matrix notation (space domain) the process may be written[18]

$$\hat{f} = (H^T H + \gamma C^T C)^{-1} H^T g \qquad (3.32)$$

This is derived by requiring that $\| C\hat{f} \|^2$ should be minimized subject to $\| g - H\hat{f} \|^2 = \| n \|^2$ where the $\| \ \|$'s are vector norms. γ is the reciprocal of the Langrangian multiplier λ used in determining the minimizing function above, and C is the constraint. As an example, if C is the tridiagonal matrix

$$C = \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & 0 \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ 0 & & & 1 & -2 \end{pmatrix} \qquad (3.33)$$

then the second order differences of $\hat{f}$ are minimized. This could be used to reduce the noise content of the computed $\hat{f}$. Hunt's method of implementing constrained least squares restoration involved forming extended column matrices of f, g, and n and a block circulant matrix H. These are transformed, a value of γ is chosen, $\hat{f}$ estimated and the residual $(g-H\hat{f})$ calculated. A different γ is chosen and the corresponding $\hat{f}$ and residual determined. This allows the prediction of a better γ, by a Newton-Raphson-like procedure, which is used in the next iteration. The γ satisfying the constraint above was found typically in 4 - 7 iterations by Hunt, and could be used to get the best constrained least squares estimate $\hat{f}$.

Clemente *et al.*[19] introduce the additional constraint that the total value of the distribution should remain constant. Thus the solution becomes

$$\begin{pmatrix} \hat{f} \\ \mu \end{pmatrix} = \begin{pmatrix} H^T H & \gamma C^T C & w \\ & w^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} H^T g \\ s \end{pmatrix} \qquad (3.34)$$

where μ is another Langrangian multiplier, introduced because of the additional constraint that the total activity should equal s, and w is a vector with appropriate weights for quadrature. If a simple summation is used then the elements of w are all unity. Since one knows that an image distribution must be nonnegative, one could also introduce the constraint that the elements of $\hat{f}$ should be nonnegative[20]. However additional constraints will naturally increase the complexity and quantity of calculation so that in most cases it would not be worthwhile.

Equation (3.32) has a similar form to the generalized Wiener filter. If ϕ_n and ϕ_f are the noise and signal covariance matrices respectively

$$\phi_f = E\{(f-\bar{f})(f^*-\bar{f}^*)^T\} \qquad (3.35)$$

where E{ } denotes the expectation value, then the generalized Wiener filter is[21]

$$\phi_f H^T (H\phi_f H^T + \phi_n)^{-1} \qquad (3.36)$$

which for $g = s + n$, *i.e.* $H = I$, reduces to the form given by Pratt[22]

$$r = \phi_s (\phi_s + \phi_n)^{-1} \qquad (3.37)$$

Pratt's method of implementing the generalized Wiener filter (3.37) for noise reduction is to transform the vector g and filter r (for a one-dimensional filtering operation; extension to two dimensions is simple)

$$G = Tg$$
$$R = TrT^{*T}$$

and obtain the optimal estimate $\hat{s}$ of s from the inverse transform of the product:

$$\hat{s} = T^{-1} RG$$
$$= T^{-1} RTg$$

But for a unitary transform $T^{*T} = T^{-1}$, so that

$$T^{-1} RT = T^{-1} TrT^{-1} T = r$$

and

$$\hat{s} = rg$$

so it seems rather pointless to transform the quantities in the first place. The total number of operations are[22] N^2 for the identity transform, $2N^2 + N$ for the Karhunen-Loéve transform (since the filter matrix R is diagonalized by the Karhunen-Loéve transform) and $N^2 + 2N\log_2 N$ for the Fourier and Hadamard transforms. However the number of filter multiplications required may be reduced

by setting all except the largest elements to zero (these occur on and near the diagonal). Thus the Hadamard transform may become competitive with the identity transform since Pratt states that the former concentrates the large values more effectively along the diagonal. He provides examples of images processed by division into 16×16 submatrices which are then Wiener filtered using the Hadamard transform.

Another restoration method based on the assumption that an image is a probability distribution has been described by Richardson[23]. It makes use of Bayes' theorem which states that

$$P(f_i|g_k) = \frac{P(g_k|f_i)P(f_i)}{\sum_j P(g_k|f_j)P(f_j)} \qquad (3.39)$$

where $P(a|b)$ is the probability of an event a occurring, given that event b has occurred. In an image restoration context this may be interpreted as the probability of an activity f at point i of the image given the value of the activity g at k of the degraded image.

By manipulating equation (3.39), Richardson obtains the iteration equations

$$f(i,j+1) = f(i,j)\sum_k \frac{h(k-i+1)g(k)}{\sum_l h(k-l+1)f(l,j)}$$

with (3.40)

$$f(i,1) = \sum_k \frac{h(k-i+1)g(k)}{\sum_l h(k-l+1)}$$

for a one-dimensional process. The two-dimensional forms are the same except that summation occurs over two subscripts instead of one. Richardson presents examples of restored images which had been degraded by an impulse response and then corrupted with multiplicative noise.

He claims that the process produces better quality restorations than a least squares process, though one should bear in mind that his technique requires much more computation.

Nahi and Assefi[24] have presented a method of recursive filtering using Kalman filtering[25 26] techniques. An image is represented as a one-dimensional function obtained by positioning the rows sequentially. The autocorrelation function of this representation is thus of a periodic nature. A difference equation model is constructed with a solution whose autocorrelation function is approximately the determined autocorrelation function and a recursive form of the model is utilized to implement the estimator enabling one-step predictions of the image. In practice the estimate is performed in both directions along the function and the mean taken as the final estimate. The model assumes that the image is corrupted with additive white noise.

Habibi[27] has proposed a two-dimensional version of the above recursive technique which has advantages of eliminating the nonstationary effects of raster scanning, and of reduced computation. It depends however on the existence of a separable autocorrelation function for the image. It has been found experimentally that a good approximation for the autocorrelation function for many images is given by

$$\phi(x,y) = \sigma^2 e^{-\alpha|x|-\beta|y|} \qquad (3.41)$$

which satisfies the separability requirement.

A further improvement has been described by Jain and Angel[28]. The advantages of their method are that the filter is isotropic, in contrast with those of Nahi and Habibi, and it allows fast implementation of the filter, whose estimate is close to that of the optimal interpolator, by using the FFT.

Let us consider one more example of space-invariant restoration before mentioning techniques for tackling space-variant systems. Linear motion blur may be expressed by (3.3) with h a symmetric rectangular function whose width equals the length of the blurring motion. Thus the transfer function is sinc(u). Now an attempt at inverting this degradation by simple division by H would fail owing to the presence of zeroes. A Wiener filter could be used, *viz.*

$$R = \frac{\sin u}{\sin^2 u + u^2 B} \qquad (3.42)$$

where $B = \phi_n / \phi_f$ and is often taken as constant or some smooth monotonically varying function, depending on the estimate of signal and noise spectra. For a long blur, the sin functions will oscillate rapidly so that it is most unlikely that a good restoration will be achieved even if noise levels are comparatively low. Sondhi[21] has mentioned an alternative approach using differentiation: for $n = 0$ in (3.3) we find for a blur in the x direction that

$$g(x) = \int_{-\infty}^{\infty} h(x-u) f(u)\, du$$

$$= \int_{x-a}^{x+a} f(u)\, du$$

for a blur of length $2a$. Differentiating with respect to x gives

$$g'(x) = f(x+a) - f(x-a) \qquad (3.43)$$

He then gives the following discrete version

$$f(x+(k+1)a) = g'(x+ka) + f(x+(k-1)a) \qquad (3.44)$$

However (3.44) is not strictly correct for we can easily show that the exact restoration in discrete form is accomplished by using not g' but a difference between successive pairs of elements of g:
Let

$$g(j) = \sum_i f(i)h(j-i)$$
$$= \sum_{j-n}^{j+n} f(i)$$

if $h(i) = 1$ for $i = j-n,\ldots,j+n$
Thus

$$g(j) - g(j-1) = \sum_{j-n}^{j+n} f(i) - \sum_{j-1-n}^{j-1+n} f(i)$$
$$= f(j+n) - f(j-1-n) \qquad (3.45)$$

If the width of the nonzero part of f is less than $2a$ then $f(x+a)$ and $f(x-a)$ will be completely separate and f may be recovered directly. Even if they overlap they may be recovered by the recursive formula above, provided that at least one nonoverlapping point is known at one end of the blur. In practice however the presence of noise would make such a restoration difficult.

We shall consider the processing of a nonoverlapping case, and analyse the blur in continuous space, since it is equivalent to the discrete approach provided that we make the necessary changes indicated above. Consider equation (3.43). We know that $f(x)$ is nonnegative within a region $2\beta < 2a$ say and zero elsewhere. We may assume without loss of generality that the nonnegative part of f extends equal distances to either side of the origin, *i.e.* from $-\beta$ to $+\beta$. Thus in the interval $(-\infty,-\beta-a)$, g' is zero and g is zero; in $[-\beta-a,\beta-a]$, $g' = f \geq 0$ and g is monotonic increasing from 0 to g_{max} say; in $[\beta-a,a-\beta]$

$g' = 0$ and $g = g_{max}$; in $[a-\beta, a+\beta]$, $g' = -f$ and g is monotonic decreasing from g_{max} to 0, and finally $g = 0$ in $(a+\beta, \infty)$. Defining

$$\phi_1(x) = g(x+a)$$

$$\phi_2(x) = g(x-a) \qquad x \text{ in } [-\beta, \beta]$$

we see that

$$\phi_1(x) = \int_{-\beta}^{x} f(x)\,dx + g(-a-\beta)$$

$$\phi_2(x) = \int_{-\beta}^{x} -f(x)\,dx + g(a-\beta)$$

thus

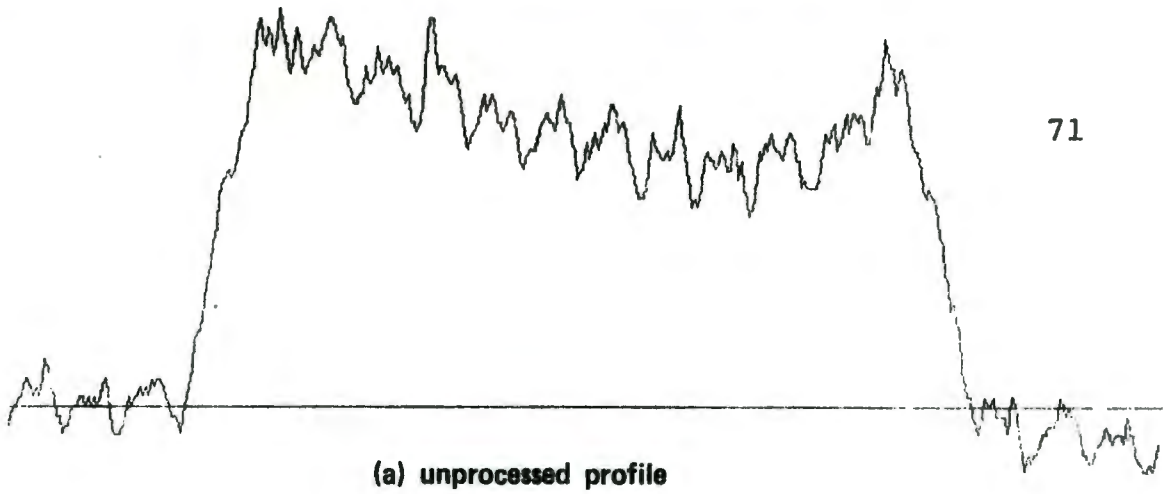
$$\phi_1(x) = \int_{-\beta}^{x} f(x)\,dx$$

$$\phi_2(x) = g_{max} - \int_{-\beta}^{x} f(x)\,dx$$

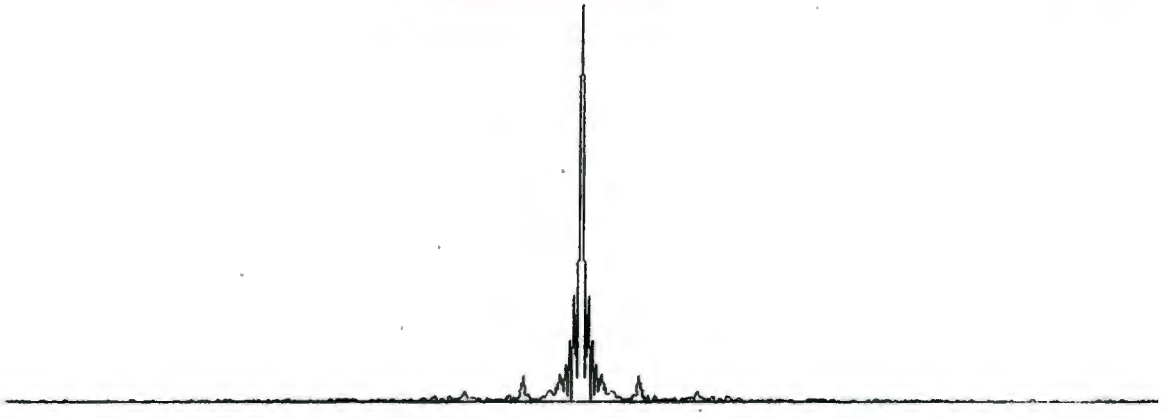
and so

$$\phi_2(x) = g_{max} - \phi_1(x) \qquad (3.46)$$

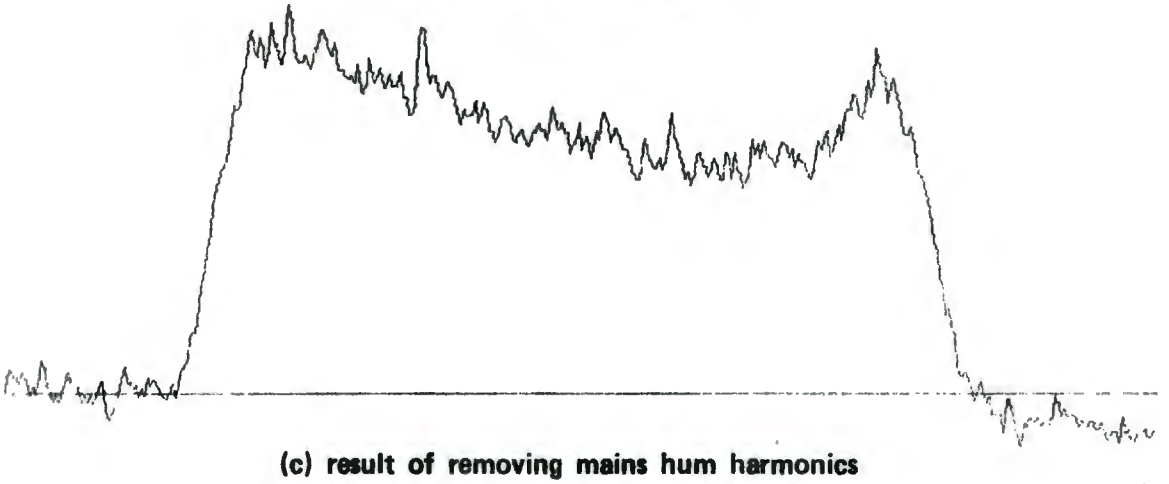
We shall now describe the method adopted in restoring a linear motion blurred photograph of the moon. See Fig. 10 and 11. Fig. 10 is a profile of the digitized image near the top end of the blur, while Fig. 11 is in the centre, so that its actual maximum is much greater although they have been scaled to the same height in the graphs. Note the presence of mains ripple, especially in Fig. 10 where the S/N ratio is much lower. All processing was done row by row unless otherwise stated. A row was Fourier transformed with the FFT and the 26^{th} and 52^{nd} spatial frequencies (visible in the spectrum)



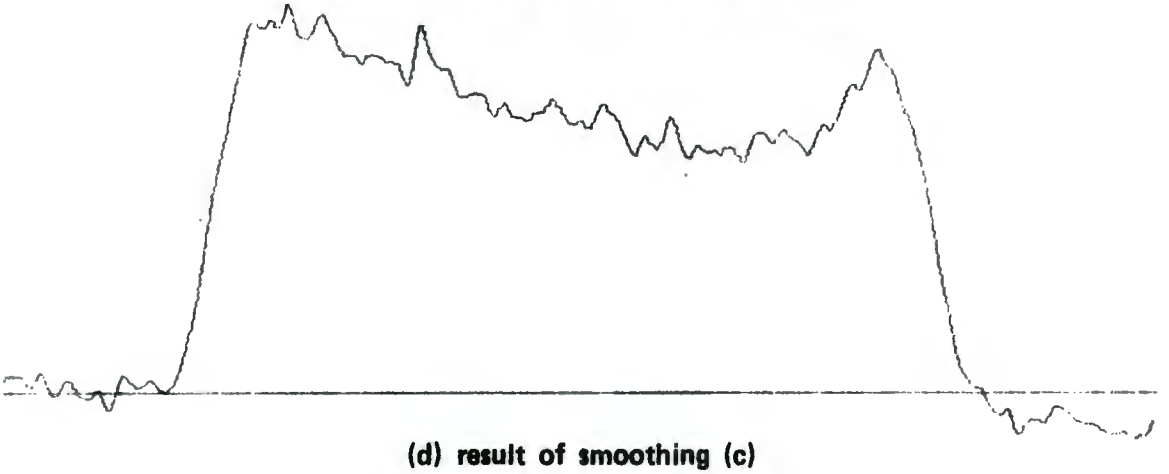
(a) unprocessed profile



(b) spectrum of (a)

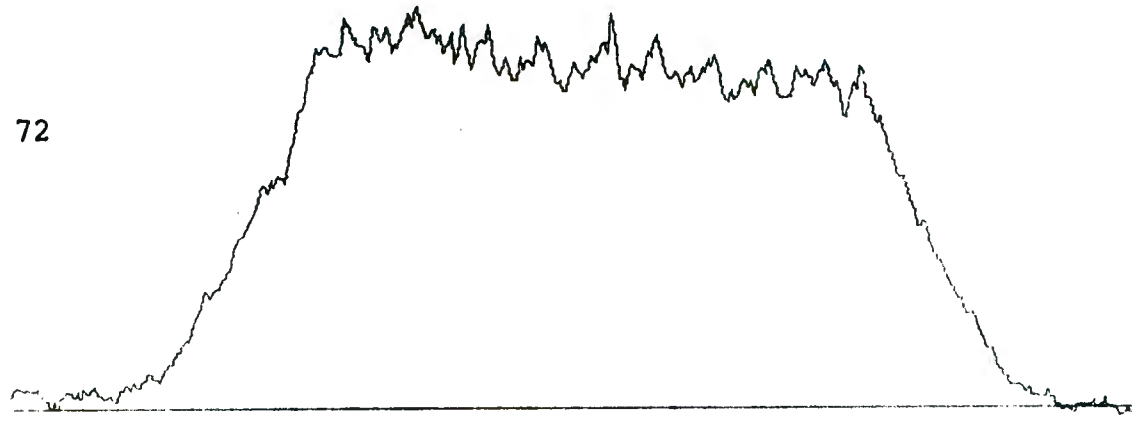


(c) result of removing mains hum harmonics

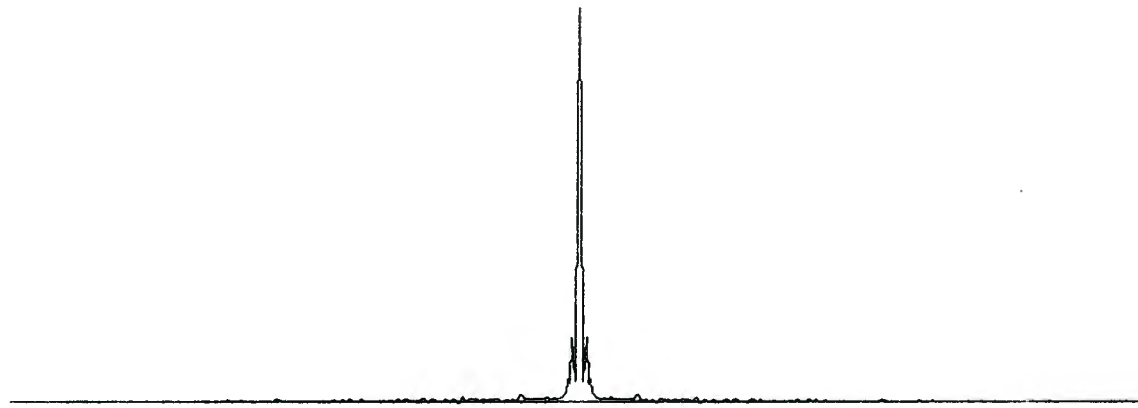


(d) result of smoothing (c)

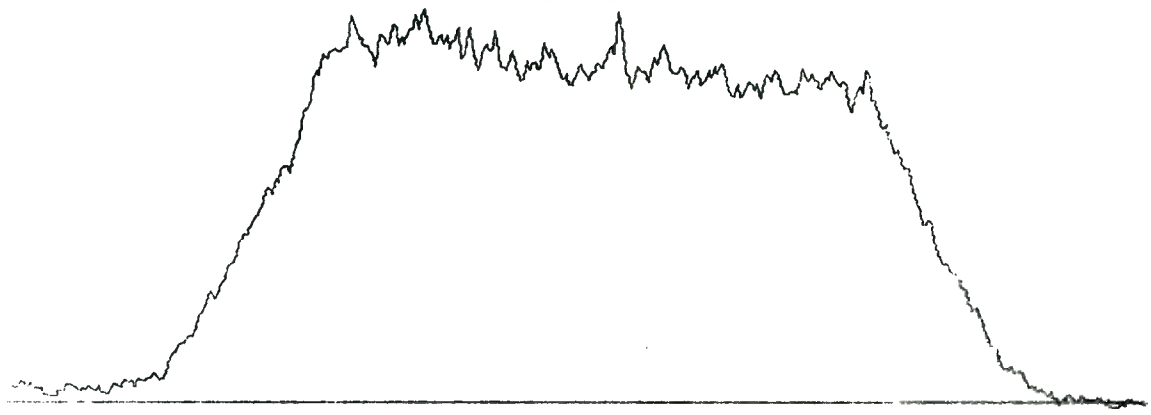
Figure 10: Profile Near Edge of Moon Blur



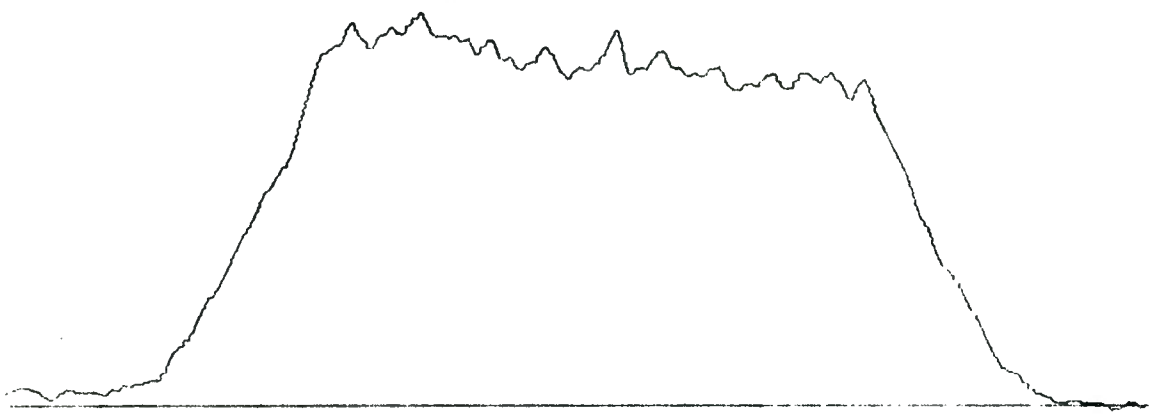
(a) unprocessed profile



(b) spectrum of (a)



(c) after mains hum removed



(d) after smoothing (c)

Figure 11: Profile Near Centre of Moon Blur

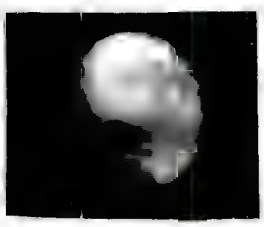


Plate 25:

Demonstration of restoration of an image degraded by linear motion blur.

Top: *clear image of moon.*
Middle: *blurred image of moon formed by taking a four-minute time exposure.*
Bottom: *restored image obtained by digitizing the blurred image and processing the resultant array digitally.*

were set to zero. (The rows comprised 512 points of which the first 400 were used.) Then the spectrum was multiplied by a Gaussian function, the DFT of $e^{-x^2/4}$, and inverse transformed. The third and fourth graphs in each figure are the image with the mains hum spectral components zeroed, before and after smoothing. Note how well the end slopes have been smoothed. The mean of the central high region was determined and all parts of the image greater than 0,95 or less than 0,05 of the mean were set equal to the appropriate bound. This was to remove the effects of noise and spikes, several of which were present in the image. As shown above, all the information is situated in the (ideally) monotonically increasing and decreasing ends of the blur. Thus the only information lost by the truncation will be at the top and bottom of the slopes, which will result in truncation of the left and right sides of the deblurred image, by about 5%. At this stage the right side of the blur was subtracted from the upper bound so that it should be identical with the left slope (equation 3.46) and shifted left until the square difference was minimized. This was performed for all rows and the mean shift determined. (The scatter was approximately 2 points either side of the mean.) Using this shift the mean difference between adjacent points of the two slopes was determined and stored in rows of 256 points. The resulting two-dimensional image was smoothed in the frequency domain (horizontally and vertically) with the same smoothing function as before *viz.* $e^{-r^2/4}$, and scaled for printing in the microdensitometer. A clear photograph of the moon, as well as the blurred and restored images are shown in Pl. 25.

Space-variant degradations cannot be processed directly using Fourier inverse techniques since space-invariance is implicit in the product-convolution re-

lationship. There are three main techniques for accomplishing restoration in such situations. The first is to attempt a deconvolution in the spatial domain with a variable impulse response. The main disadvantage is the enormous amount of data that must be stored (h is N^4 for an N×N image in general though if h is narrow, say M×M then its storage requirements reduce to M^2N^2). If h varies fairly slowly over an image then the image may be subdivided into areas in each of which h is substantially constant so that space-invariant techniques may be used. The third method involves determining a coordinate transformation which will distort the image in such a way as to make h space-invariant, and after inverting h, the inverse coordinate transformation is applied. An example illustrating this method is presented by Huang and Robbins[29]: an image obtained from an optical system with coma can be shown to have the property that the transformed image is the product of the transformed impulse response h and ideal image f provided that the images are expressed in cylindrical coordinates (r,θ) and the transformation consists of a Fourier transform with respect to the angular variable θ and a Mellin transform with respect to the radial variable r. Now the Mellin transform of $g(r)$ is

$$\int_0^\infty g(r)r^{s-1}ds \qquad (3.47)$$

Putting $r = Ae^{\rho}$, $s = j2\pi f$, so that $r]_0^\infty = \rho]_{-\infty}^\infty$, equation (3.47) becomes

$$A\int_{-\infty}^\infty g(e^{\rho})e^{j2\pi f\rho}d\rho \qquad (3.48)$$

which we recognize as a Fourier transform. Hence we are now effectively performing a two-dimensional Fourier transform; in other words by compressing the radial dimension of the image logarithmically, it has been re-

duced to a space-invariant system. Having inverted the effects of h, by Wiener filtering for example, the restored image is obtained by expanding radially on an exponential scale. Sawchuck has presented a more general analysis of the use of coordinate transformations in reducing a space-variant system to space- invariance[30 31].

A technique for determining geometrical distortions has been published by Lillestrand.[32] It concerns the determination of positional deviations of one image relative to another so that their details may be aligned accurately after which a difference image is formed. This has an application in change detection using side looking radar images for example. One image is divided into a column of small squares which are shifted until the statistical correlation coefficient

$$\rho = \frac{\Sigma (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\Sigma (a_i - \bar{a})^2 \Sigma (b_i - \bar{b})^2}}$$

is minimized, where a_i and b_i are corresponding points in the subregions of images A and B. This allows the determination of a displacement vector for aligning each square. Having completed one column (the alignment of all squares in a column is performed in parallel to increase speed) the process is repeated for the next, and so on. Thus the required geometrical distortion to transform one image into another is completely specified.

Finally let us consider the processing of multiplicative images *e.g.* multiplicative noise

$$g = s.n \qquad (3.49)$$

Then by taking the logarithm of the image

$$\ln g = \ln s + \ln n$$

or

$$g' = s' + n' \qquad (3.50)$$

which is identical in form with (3.5) and may be processed similarly to obtain $\hat{s}'$ so that

$$\hat{s} = e^{\hat{s}'} \qquad (3.51)$$

Oppenheim *et al.*[33] have used this method in images which may be regarded as the product of a low frequency illumination i and high frequency reflectance r

$$f = i.r \qquad (3.52)$$

Thus by taking the logarithm of the image the two components may be separated and filtered independently, assuming that their spectral bands do not overlap very much, which is usually a valid assumption for the reason given above. However this technique lies equally in the field of image enhancement and will be mentioned again in the next chapter.

In this chapter we have dealt exclusively with digital techniques with the exception of raster removal. It is possible however to implement some inverse filters in a coherent optical system. The filters are in general required to perform phase as well as amplitude modifications on an image spectrum, and may be constructed holographically or by digital means[34], the two main methods of implementing the latter being due to Lohmann and Lee. In both cases the filter is divided into an array of rectangles, in each of which a transmission aperture whose area is proportional to the magnitude of the filter function at that point and whose position in the rectangle is chosen to make the path difference relative to the other rectangles between the filter and

the first order diffraction patch correspond to the desired phase. In Lohmann's method, the aperture is shifted the desired distance in the rectangle. In Lee's method the rectangle is divided into four areas, the first and third corresponding to positive and negative real values, the second and fourth to positive and negative imaginary values. The transmision apertures are placed in the appropriate areas, at most one of the real and one of the imaginary areas. Thus Lohmann's method is analogous to writing a complex number as $re^{i\theta}$ whereas Lee's method corresponds to $(a+jb)$ notation. A program was written to draw Lee holograms on a Calcomp plotter. Three examples, *viz.* a square, a one-dimensional differentiation filter and a two-dimensional differentiation filter, are shown in Pl. 31. As mentioned before it was considered more worthwhile to use digital techniques owing to their greater flexibility and superior performance in general, so that optical processing has been glossed over. Articles by Stroke[35] and vander Lugt[36] describe some optical techniques.

In concluding this chapter we may mention the use made of so-called *a priori* and *a posteriori* information available in attempting to implement restoration techniques. *A priori* knowledge is obtained without reference to the degraded image, for example the impulse response of an imaging system determined by measurements on the system, the presence of motion blur or defocussing during image formation, geometrical distortions as a result of camera tilt, the statistics of the type of noise that will be present in an image (*e.g.* film grain in photography or quantum fluctuations in scintigraphy), and the fact that images are nonnegative and imaging systems approximately energy conserving. *A posteriori* knowledge on the other hand is gleaned from the image itself. Examples could be the estimation of the impulse response from distributions in the degraded

image which are known to be point sources in the ideal image, the determination of the length of motion blur by direct measurements on an image or by deduction from the periodicity of the logarithm of the Fourier transform of an image in the direction of blurring (cepstrum analysis), an estimation of noise from fluctuations in areas of the image which should be smooth, or from regions of the spectrum where the image spectral amplitudes should be small or zero. Clearly each situation will require special attention if an optimum restoration is desired. The recovery of a deblurred image of the moon presented earlier in the chapter illustrates some of these points. One can see that the variety of modifications that can be incorporated into standard techniques by taking advantage of *a priori* and *a posteriori* information is limited only by the ingenuity and perceptiveness of the person involved.

Obviously a generalized process (primarily *a priori* though algorithms could be developed to enable the processing system to evaluate certain information from the image being processed) could be developed for a given type of degradation permitting automatic restoration of images of that particular kind. This would be essential in a routine utility environment such as a hospital where a technician would monitor a system, instead of a highly trained specialist in image processing.

IMAGE ENHANCEMENT

CHAPTER 4

While the aim of image restoration is clear and well defined, image enhancement is a rather nebulous concept, mainly because it is so dependent on the particular situation in which it is to be applied - enhancement processes in two different cases might be inverses of each other. In an attempt to define it we could say that image enhancement involves modifying an image distribution so as to present the desired image information in such a way that the viewer will interpret it correctly and glean most relevant information. Thus in some cases *e.g.* commercial TV pictures, the enhanced image will generally appear very similar to the restored image from which it is derived, whereas an X-ray image might be considerably modified to present only certain aspects of the image such as the outlines of an object. In diagnostic applications such as the latter, an enhancement process should remove as much irrelevant information as possible, which is merely a distraction, to facilitate and speed up the diagnostician's work.

In setting about enhancing an image then, the task is to decide what image information is required, how it can be presented most clearly and finally how this form should be modified to compensate for non-linearities in vision so that the brain will actually visualize the enhanced image exactly as the desired form.

Let us first consider some of the properties of human vision. One of the most well known characteristics is sensitivity to contrast. A grey object surrounded by black will appear much lighter than the same object with a white background. Thus if it is desired that both should appear to be the same shade of grey, then the former should be darkened, the latter lightened.

Similarly the brain is much more sensitive to rapid intensity changes than to slow variations. As a result the average shade of an image may vary by as much as 50% without being noticeable[37]. A phenomenon directly associated with these properties is the Mach effect (see Pl. 26). At the boundaries between the grey scale rectangles, there is a sharp discontinuity so that the light edge appears lighter than it actually is and

(a) actual profile *(b) observed profile*

Fig. 12: Mach effect (see Pl. 27)

vice versa (see Fig. 12). However the centres of adjacent rectangles appear to be of very similar shades of grey since the eye is relatively insensitive to contrast changes over the larger distance. Hence if it is desired that the grey scale should be seen as rectangles of uniform density then the contrast at the

(a) actual profile *(b) observed profile*

Fig. 13: Profile of Fig. 12 after compensation for non-linearities of the visual system.

edges should be reduced *i.e.* the corners of the density profile should be rounded (Stockham[38] has published pictures illustrating this). Hence low frequency variations in an image may be reduced in amplitude, thus allowing

Plate 26:

Top: *grey scale*
Middle: *lung and Picker thyroid images after five restoring iterations.*
Bottom: *lung and Picker thyroid images restored by density shifting.*

local contrast to be increased. This is beautifully illustrated by pictures in the paper by Oppenheim *et al.*[33] Related to this is the fact that edges are important - reduction of contrast at edges results in a "fuzzy" picture.

Another related effect is that of estimating the apparent brightness of a fairly uniform area by the contrast or strength of its borders[37]. An illustration of this is how well a relatively complex picture may be represented by a simple outline drawing.

The sensitivity of human vision as a function of spatial frequency has been determined by Lowry and De Palma[39], and the response was shown to peak in the middle frequency range, *i.e.* the eye is insensitive to low frequencies, and at high frequencies, when the resolution limit is approached, the response must naturally decrease as well.

Temporal sensitivity is really not relevant to a discussion of static images, but it is of interest to note that the response curve has a similar shape to that for spatial frequencies, and peaks at 5 - 15Hz, with a gain of up to 10dB over the very low frequency response.

Schreiber also mentions the visibility of noise in an image. The most important characteristics are that noise is less visible if it is randomly distributed or if it occurs in areas of an image which have a large amount of detail, and that its presence drastically reduces contrast and consequently sharpness in images.

Having gained some insight into the requirements for enhancing an image we may now consider various techniques for accomplishing the desired results. It should be emphasized that the scope for developing and utilizing different methods is enormous.

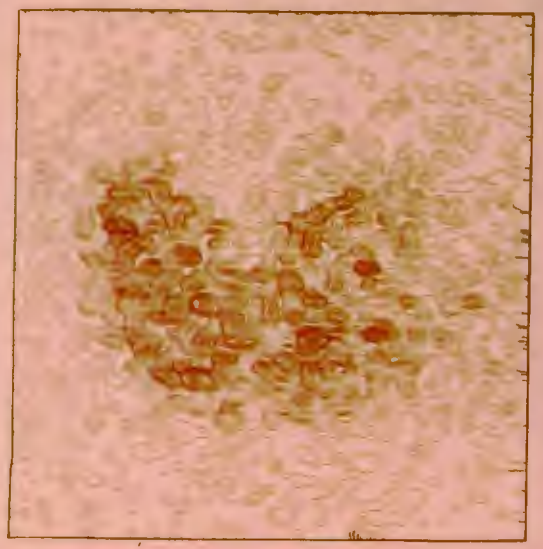
One very successful approach is that of nonlinear

filtering, mentioned above (equation (3.51)). Since the illuminance component i is essentially low frequency, while the reflectance component r is determined largely by detail in an image and consequently mainly in the medium to high frequency range, the spectrum of the logarithm of the image separates the two components comparatively successfully. As a result the illumination component may be attenuated without affecting detail in an image and the processed image has greater local contrast so that it appears crisper. Furthermore by removing low frequency intensity variations to which the eye is insensitive, the remaining contrast variations are expanded to make full use of the grey scale range so that image information is more readily assimilated.

Another nonlinear process which may be used to increase image contrast has been proposed by Andrews *et al.* [40] It involves transforming an image f by a Fourier or Hadamard transform, whose magnitude is then raised to a fractional power, while preserving phase ($e^{i\theta}$ for the Fourier transform, $e^{ij\pi}$ for the Hadamard since ± 1 are the only possible values for the phase of a real function). This may be written as

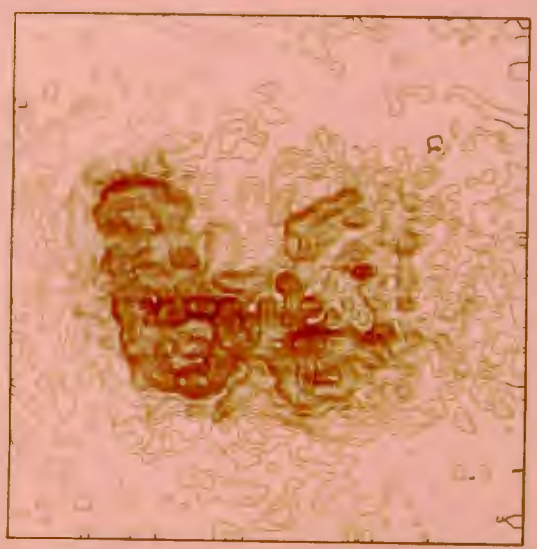
$$f \rightarrow F \rightarrow |F|e^{i\theta} \rightarrow |F|^{\alpha}e^{i\theta} \rightarrow f' \qquad \alpha \text{ in } (0,1)$$

This has been carried out on the lung and Picker thyroid images, obtained at the fifth stage of the iterative restoration technique (see Pl. 28) for $\alpha = \frac{1}{4}$, $\frac{1}{2}$, and $\frac{3}{4}$. For $\alpha = \frac{3}{4}$, the contrast is considerably enhanced while preserving the basic structure of the image. For $\alpha = \frac{1}{2}$, and to an even greater extent for $\alpha = \frac{1}{4}$, the images become unrecognizable. However it is interesting to note that large variations in the original image appear as bright spots in the processed image so that the latter is useful for locating variations which may then be studied in the $\alpha = \frac{3}{4}$ image, to determine whether they

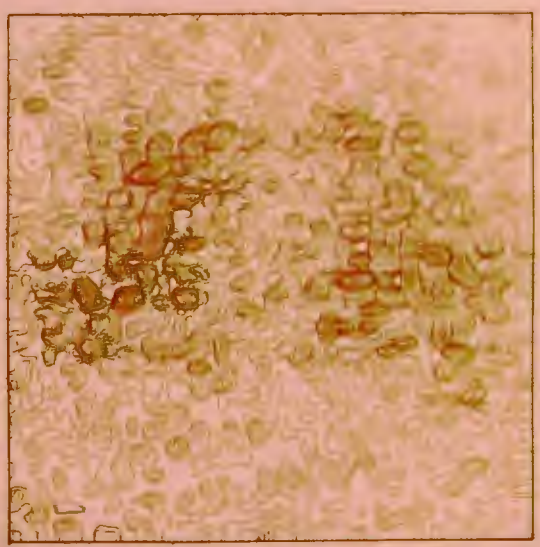


(A) NONLINEARLY PROCESSED PICKER THYROID (0,25)

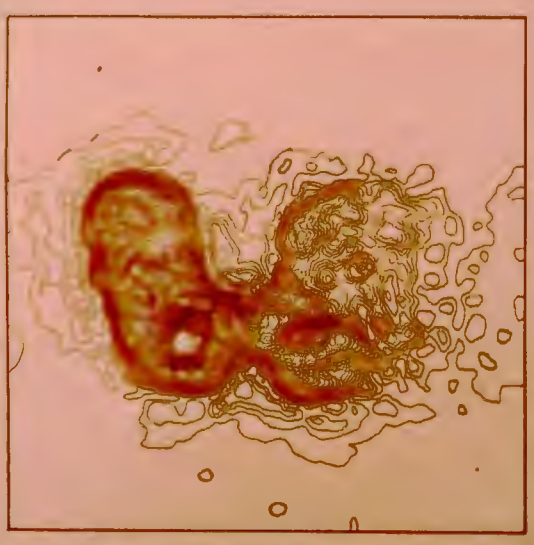
(B) NONLINEARLY PROCESSED LUNG (0,25)



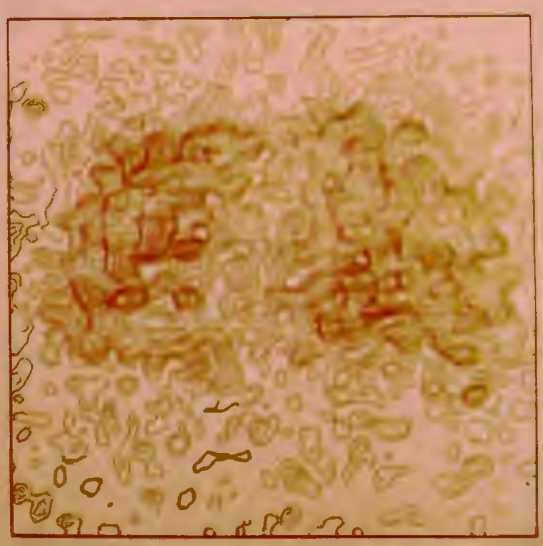
(C) NONLINEARLY PROCESSED PICKER THYROID (0,50)



(D) NONLINEARLY PROCESSED LUNG (0,50)



(E) NONLINEARLY PROCESSED PICKER THYROID (0,75)



(F) NONLINEARLY PROCESSED LUNG (0,75)

PLATE 27.

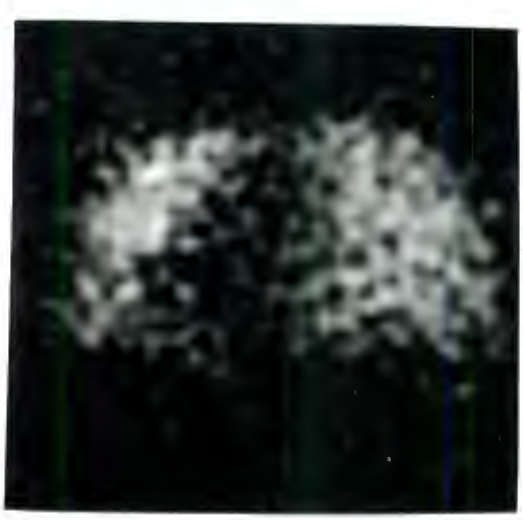


Plate 28:

Nonlinearly processed lung and Picker thyroid images (see Pl. 27).

Top: $\alpha = 0,25$
Middle: $\alpha = 0,50$
Bottom: $\alpha = 0,75$

are of clinical significance.

An intensity distribution histogram may be determined for any image. It displays the relative occurrence of each shade of grey in the image concerned. Now if the distribution is constant, then the same use is made of each band of the grey scale so that the dynamic range is used more efficiently and is likely to lead to a superior image in general. Let us determine the form of the mapping function which will equalize an arbitrary distribution.

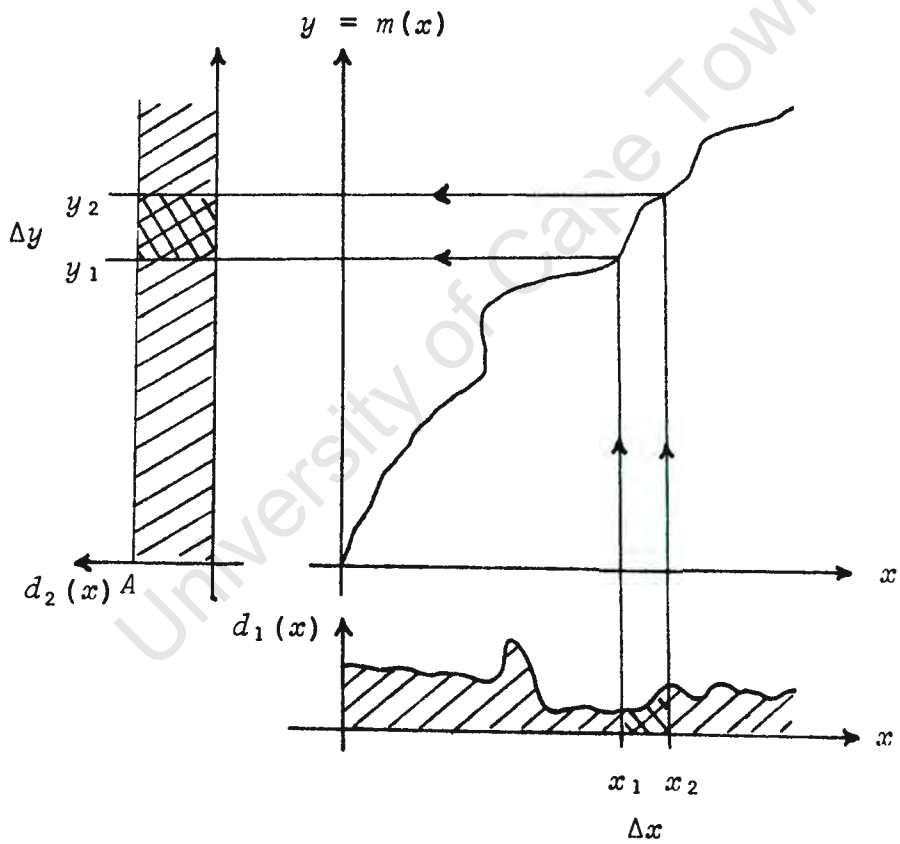


Fig. 14: Determination of mapping function to equalize histogram distribution.

Let $d_1(x)$ be the unnormalized distribution of an image f, $d_2(y)$ be the equalized distribution of the processed image f', and $y = m(x)$ be the function mapping d_1 into d_2. Let the grey scale variation from black to white

correspond to the interval [0,1]. Then x,y are in [0,1]. Let the total area of f (and consequently f') be A, so that

$$\int_0^1 d_1(x)\,dx = \int_0^1 d_2(x)\,dx = A$$

Then $d_2(x) = A$.

Consider the cross-hatched areas of d_1 and d_2. These must be equal since every point in the first is mapped into the second.

Hence

$$d_2(y_1)\Delta y \simeq d_1(x_1)\Delta x$$

and

$$\begin{aligned}\Delta y &= y_2 - y_1\\ &= m(x_2) - m(x_1)\\ &\simeq m'(x_1)\Delta x\end{aligned}$$

and thus

$$d_2(x)m'(x)\Delta x \simeq d_1(x)\Delta x$$

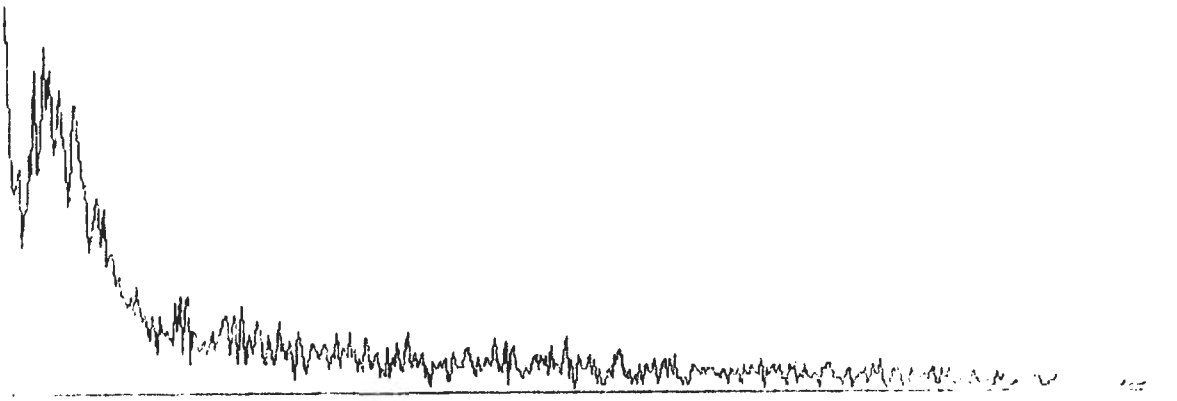
and

$$Am'(x) = d_1(x)$$

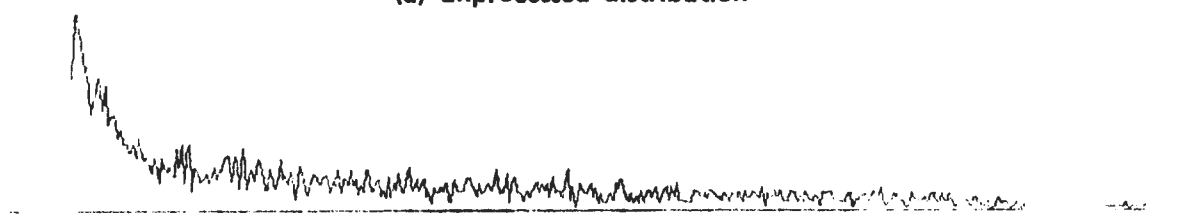
so that

$$m(x) = \frac{1}{A}\int_0^x d_1(x)\,dx$$

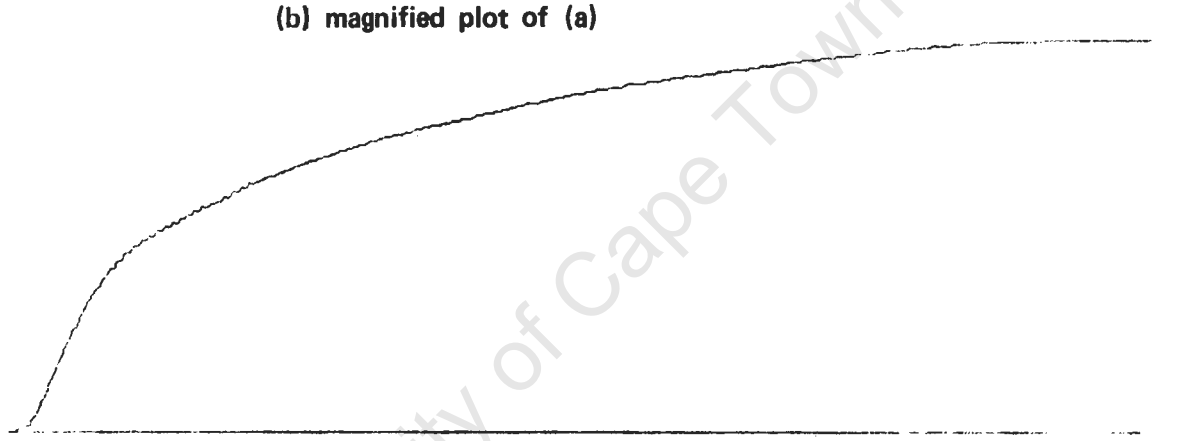
Thus the required mapping function is the normalized integral of the unprocessed intensity distribution histogram. These functions are plotted (Fig. 15 and 16)



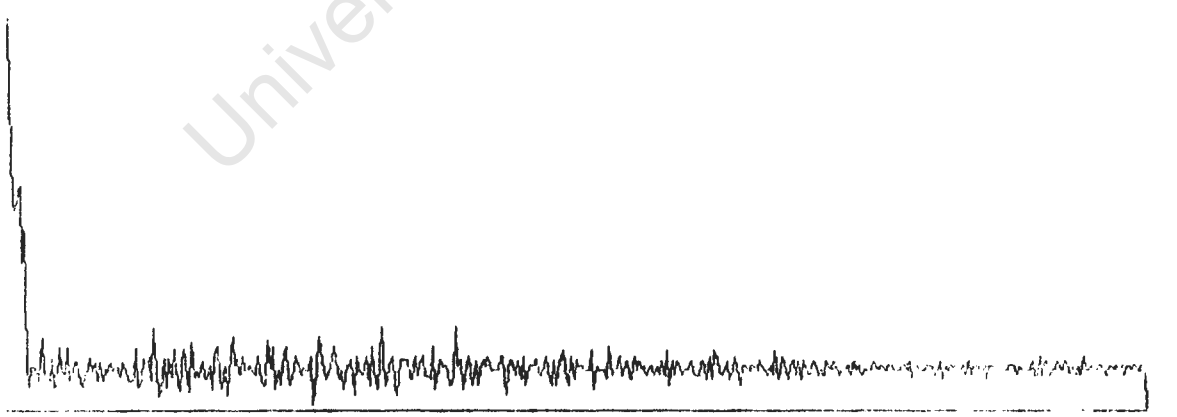
(a) unprocessed distribution



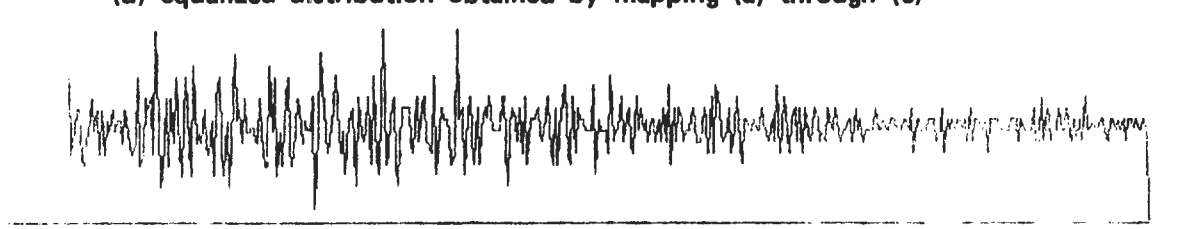
(b) magnified plot of (a)



(c) mapping function for equalizing distribution of (a) and (b)

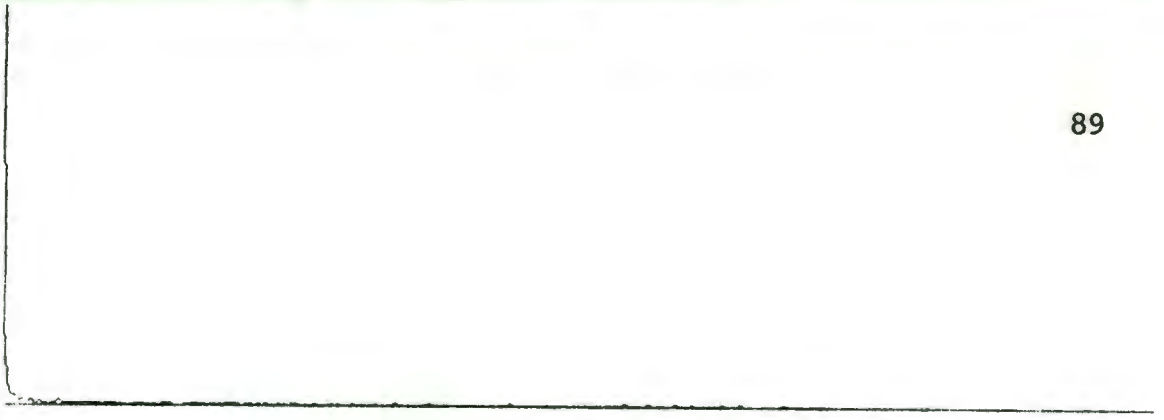


(d) equalized distribution obtained by mapping (a) through (c)

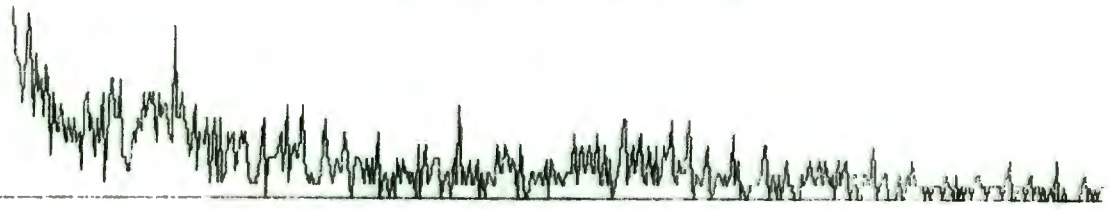


(e) magnified plot of (d)

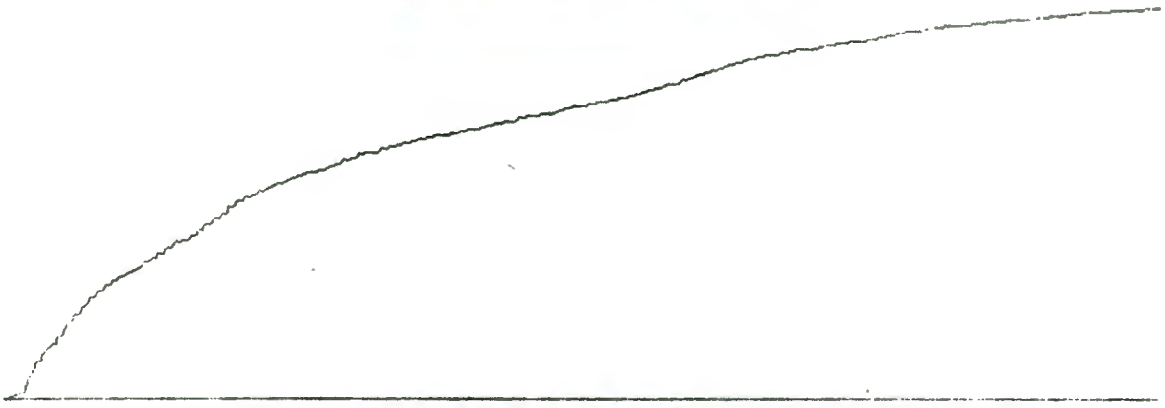
Figure 15: Equalization of Density Histogram of Restored P/A Lung (5 iterations)



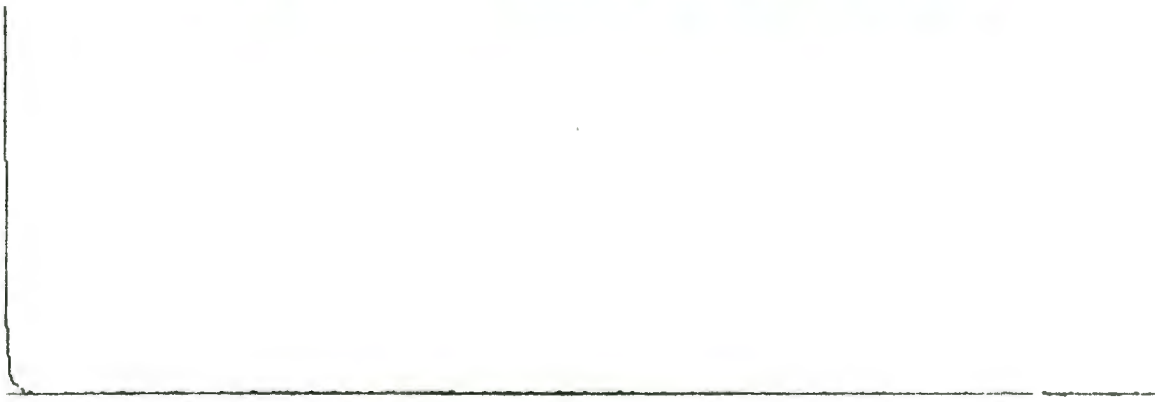
(a) unprocessed distribution



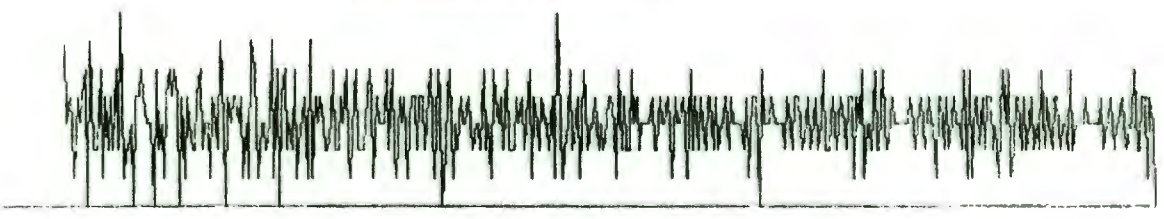
(b) magnified plot of (a)



(c) mapping function for equalizing distribution of (a) and (b)

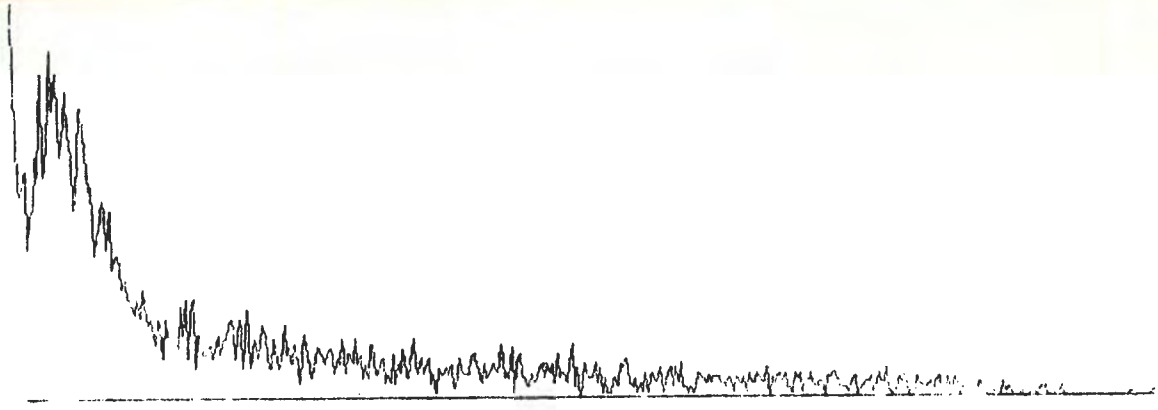


(d) equalized distribution

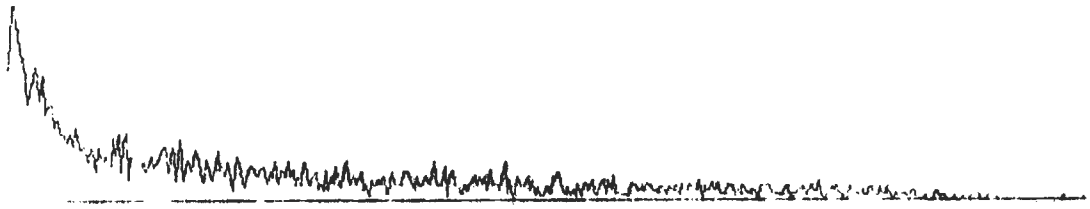


(e) magnified plot of (d)

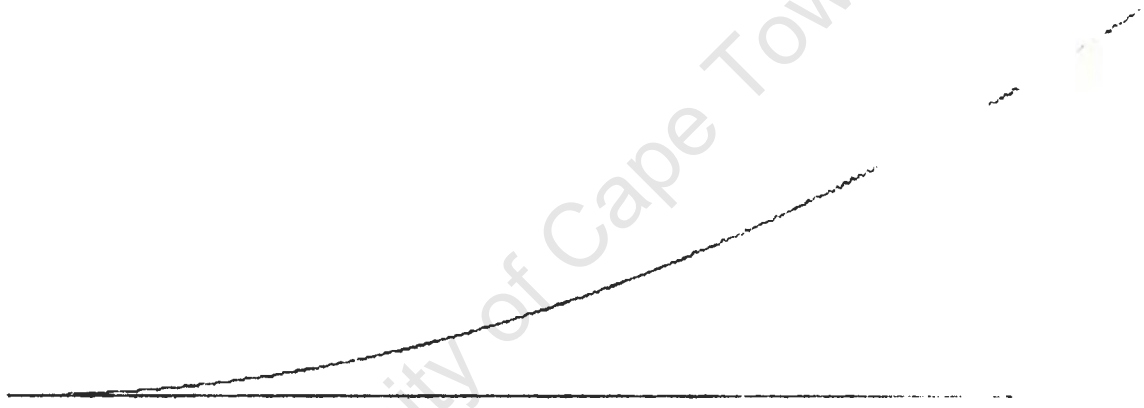
Figure 16: Equalization of Density Histogram of Picker Thyroid Phantom (rest. with 5 iter.)



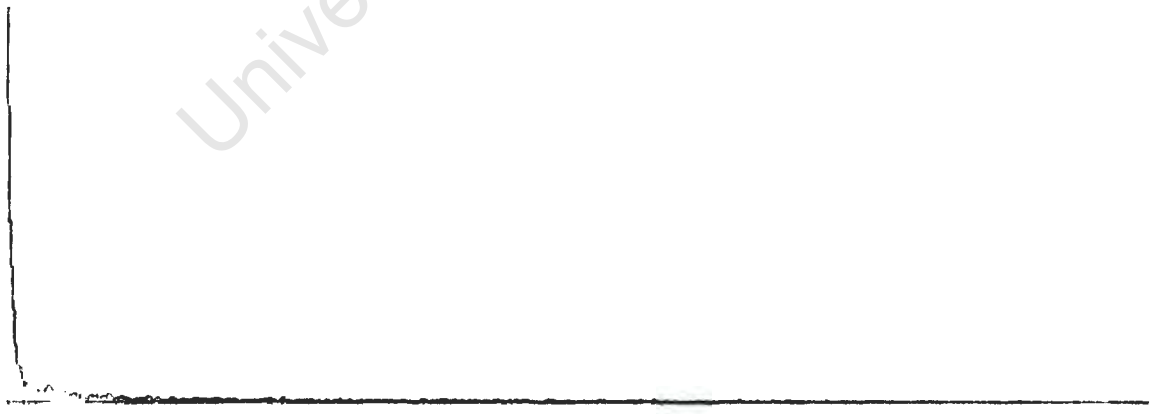
(a) unprocessed distribution



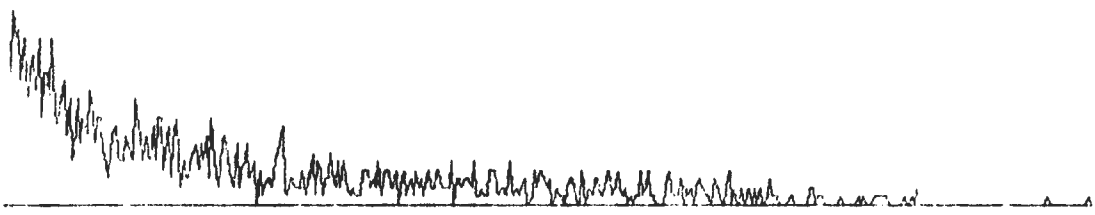
(b) magnified plot of (a)



(c) square mapping function

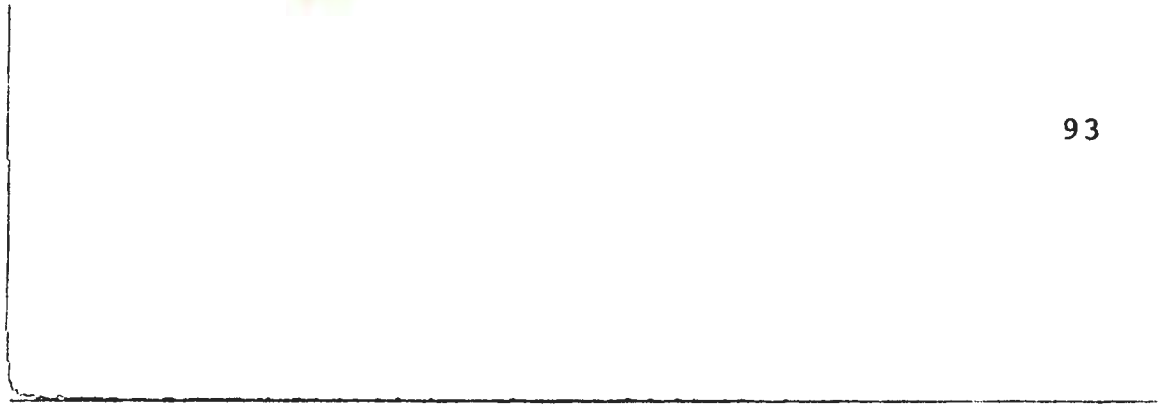


(d) result of mapping (a) through (c)

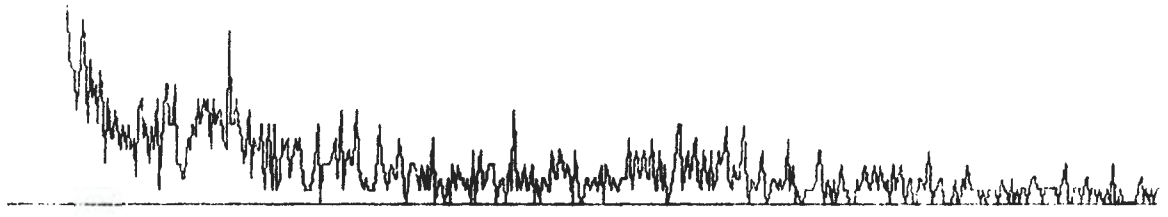


(e) magnified plot of (d)

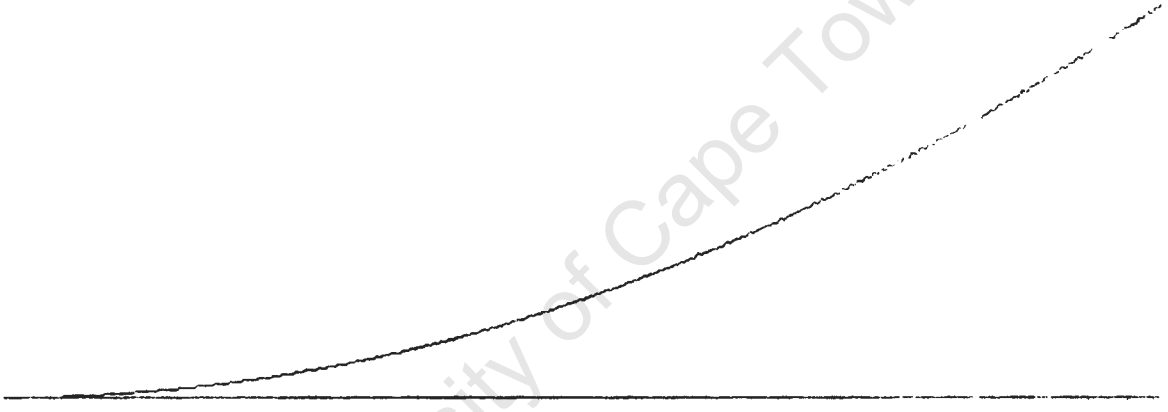
Figure 19: Square Mapping of Restored P/A Lung Density Histogram



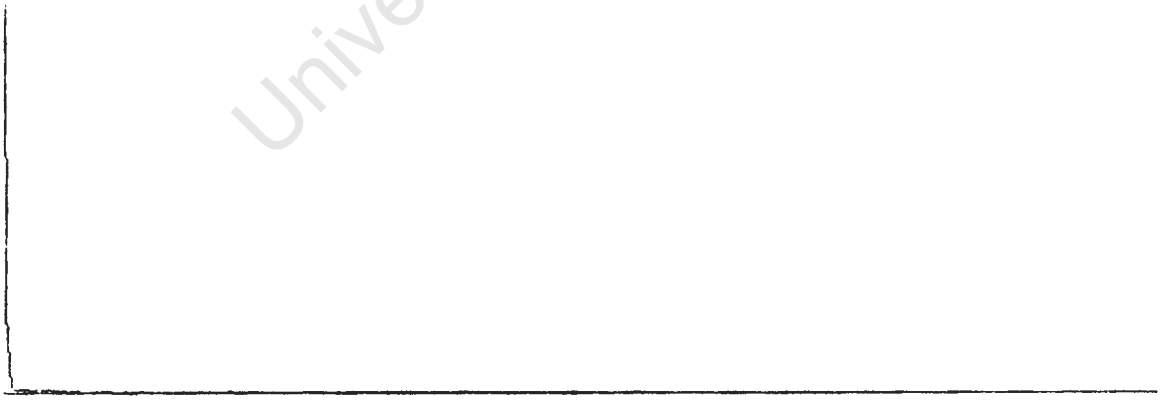
(a) unprocessed distribution



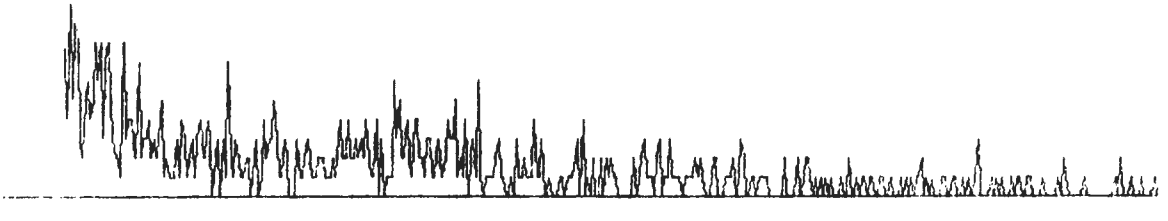
(b) magnified plot of (a)



(c) square mapping function



(d) result of mapping (a) through (c)



(e) magnified plot of (d)

Figure 20: Square Mapping of Restored Picker Thyroid Phantom Histogram

for the lung and Picker thyroid phantom images obtained from 5 iterations in the restoration technique discussed in the previous chapter. The images after histogram equalization are shown in Pl. 29. Note that the lowest 2% of the images (black background) was left unaltered since it contained no information but covered a considerable area of the image. If it had been included in the mapping the remaining levels would have been compressed into the bright region of the grey scale with an attendant loss of contrast and information. It will be noted that the histograms in both cases are heavily biased towards darker areas of the grey scale, and that the mapping function resembles a square root function (Fig. 17 and 18). Now since the histograms are skewed towards the dark regions, an equalization process must necessarily expand the lower range of the distribution so that contrast will be improved in dark areas but reduced in light areas. This effect is even more pronounced with a square root mapping function as it is much steeper in the dark shades, resulting in greater expansion (Pl. 29). A square mapping function (Fig. 19 and 20) on the other hand compresses lower shades into a small range while contrast in light areas is increased as shown in Pl. 29. Thus the choice of mapping function will be governed by the region of the grey scale in which increased contrast is most desired. For example,

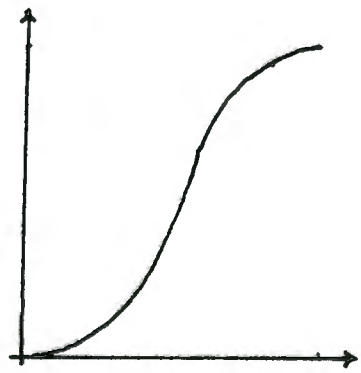


Fig. 21: Mapping function to improve contrast in midrange greys.

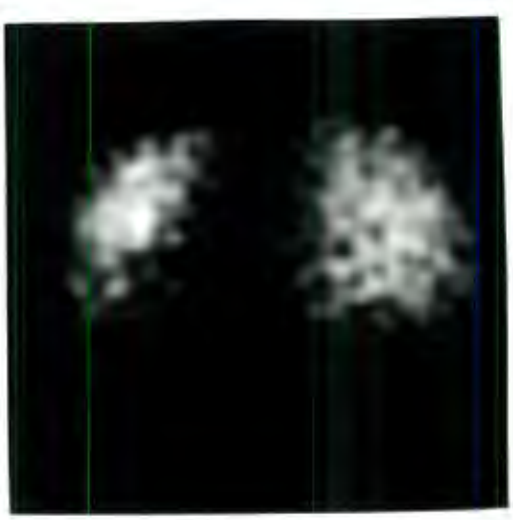


Plate 29:

Modification of grey scale distribution histograms for lung and Picker thyroid images restored with five iterations.

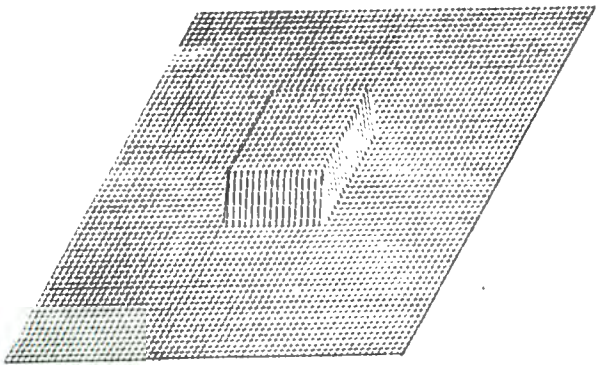
Top: *equalizing mapping function*
Middle: *square root mapping function*
Bottom: *square mapping function.*

the mapping function shown in Fig. 21 would result in improved contrast in midrange greys.

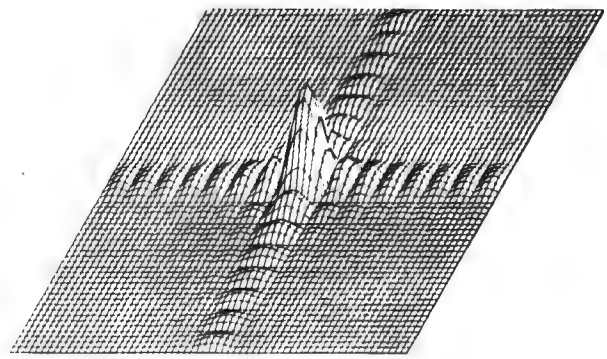
While on the subject of contrast, it may be mentioned that the use of colour displays increases the available dynamic range of a display by a factor of several hundred so that it appears to offer considerable scope in image enhancement.

A technique related to that of contrast improvement is edge enhancement. In both methods low spatial frequency amplitudes are reduced. Edge enhancement is usually a more drastic procedure where the spectrum is multiplied by a ramp (differentiation) or the low frequencies may be removed altogether. This is illustrated in Pl. 30. A square and its DFT are shown together with the edge enhanced image obtained by removing low frequency lobes. The same may be accomplished optically. The square in the top left of the bottom photograph has been high-pass filtered to produce the edge enhanced images. The cutoff frequency of the filter for the right image is higher than that for the left so that the edges and ripple are finer. For interest the result of low-pass filtering is illustrated by the top right image. A brief summary of optical contrast enhancement techniques is presented by Lipson[41] and covers phase contrast techniques (*e.g.* Zernicke) and Schlieren systems, which may be analysed by utilizing the fact that the real and imaginary components of the filtered image are Hilbert transforms of each other.

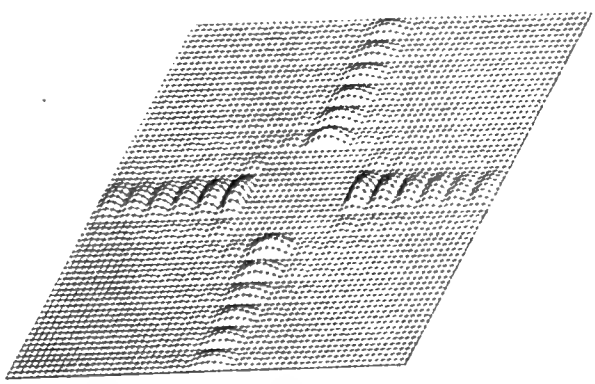
Finally we may consider various methods of displaying processed images, for the most superbly enhanced image distribution in the world is worthless if there is no suitable means of displaying it. As noted above, outlines are very important in the human visual system, which is why line drawings can convey image concepts so successfully in spite of their simplicity. By being able to "latch on" to lines in an image, the brain can more readily



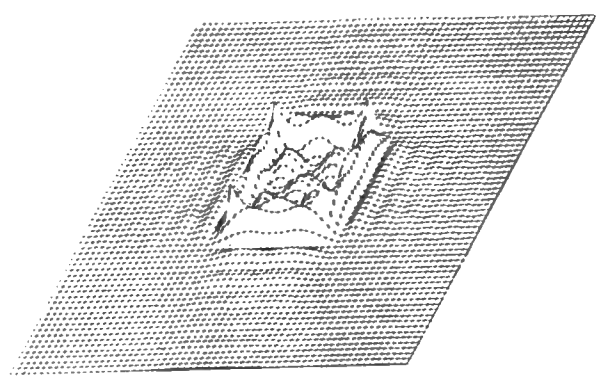
(A) SQUARE DISTRIBUTION



(B) DFT OF (A)



(C) APPEARANCE OF (B) AFTER ZEROING CENTRAL LOBES



(D) RECONSTRUCTION OF DISTRIBUTION FROM (C)

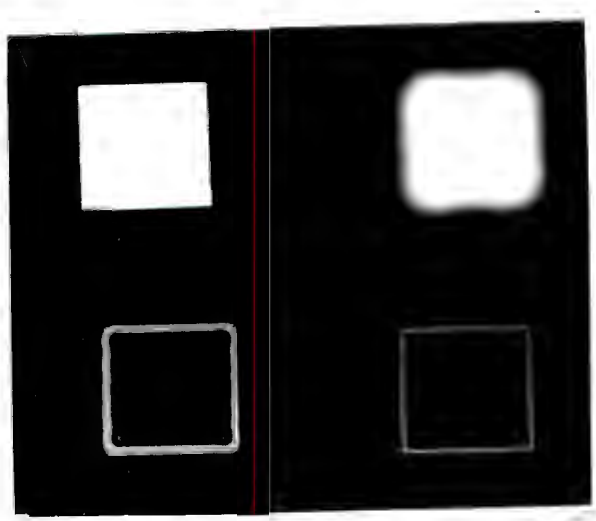
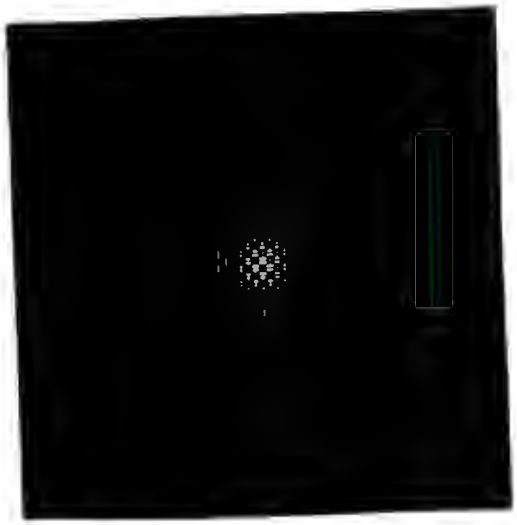
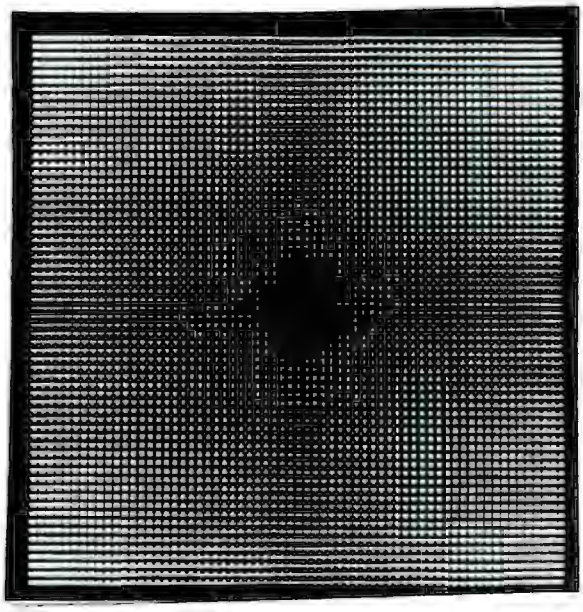


PLATE 38.





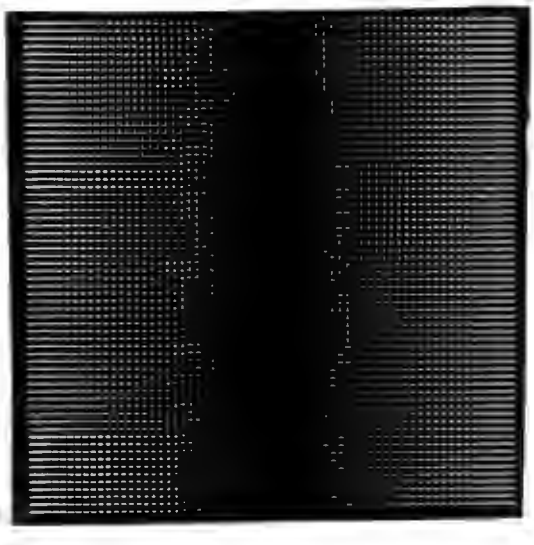


Plate 31:

Digitally constructed holograms using Lee's method.

Top: spectrum of a square distribution
Middle: two-dimensional differentiating filter
Bottom: one-dimensional differentiating filter

compare magnitudes and orientations in slowly varying regions of an image than in a continuous transition through shades of grey. This may be accomplished in a continuous image by running through the grey scale several times so that a pseudo-contour display is formed[44].

Three different formats are used in this thesis: isometric line drawings, plain contour plots and anaglyphs, and continuous photographs. For displaying a true-to-life image, a continuous photograph is essential, but for representing distributions the other forms serve adequately in many instances and have the advantage of facility of production on an X-Y plotter. The isometric displays are sensitive to local variations and consequently very poor at displaying a noisy image clearly. They do not show up very slow smooth variations as inspection of the Gaussian function of Pl. 3b will show - it appears much flatter than it actually is. Furthermore, by forming an oblique view, features in some parts of an image will be more visible than others and distortions are inevitable so that such a display should not be used where comparative studies of a quantitive nature are required. Contour plots on the other hand do not produce distortions, though a plain contour display does not convey a qualitative impression as readily as the isometric plots, and although anaglyphs can provide beautiful displays, some viewers experience difficulty in visualizing them. In addition very rapid large amplitude fluctuations in an image may be difficult to visualize (Pl. 27). It appears that a continuous grey scale picture has the greatest potential for displaying a wide variety of images in a form which can do them justice. The recent use of pseudo-colour offers great scope for increasing the quality and range of processed images. The main difficulty is finding a suitable method of forming such continuous image displays - a microdensitometer of TV screen seems to be the best solution.

It would be difficult to provide an exhaustive treatment of the subject of image enhancement for the reasons given at the beginning of the chapter. However the discussion has covered the most important concepts and indicated the objectives, requirements and scope of the field.

CONCLUSION

CHAPTER 5

The basic concepts of image restoration and enhancement have been presented and the power, flexibility and scope of digital techniques in this field should have become apparent to the reader. To convey a clearer conceptual understanding of the subject, restoration and enhancement have been dealt with separately, but one should bear in mind that in a practical situation many of the processes may be combined when determining the optimum procedure in a given situation so that having gained an insight into the field one should not compartmentalize the different areas but rather view them as a whole.

The use of unitary transforms in increasing efficiency and facility in image processing has been shown and at the moment the most outstanding appear to be the Fourier, Hadamard and slant transforms.

With regard to restoration, Wiener filtering produces good results relatively easily provided that image and noise statistics are fairly well characterized. The iteration technique is well suited to processing scintigraphic images since little if any knowledge of the noise statistics of a particular image are required, so that a standard process may be used with very little variation for a wide variety of scintigrams. In addition by providing a sequence of partially restored images, one is likely to be able to select one very close to the optimal restoration, in fact it is quite probable that it would be superior to one obtained by Wiener filtering with inadequately characterized statistics. Density shifting has been shown to function satisfactorily but its weaknesses have been indicated. It appears that recursive filtering techniques will become increasingly

prevalent in the future as they may be implemented efficiently and can also cope with space-variant situations. Although Kalman filtering theory is based on the assumption of additive white noise, its recursive implementations outlined in Chapter 3 appear very successful.

Modification of grey scale distribution histograms and contrast improvement using nonlinear filtering techniques are the most important enhancement processes, though, as has been emphasized, there is enormous scope for developing *ad hoc* processes to improve the quality of a particular image.

Distributions which do not fluctuate too rapidly may be very satisfactorily displayed with isometric plots or anaglyphs, but in general to obtain a high quality image, a continuous grey, or better still, colour display must be resorted to.

While the techniques discussed in this thesis are very interesting academically, one should bear in mind that the ultimate goal should be to develop automatic image processing systems that are commercially viable. In this connection parallel processing using recursive filtering to provide real-time processing will probably be the most feasible solution for accomplishing restoration. For enhancement, pattern recognition and artificial intelligence systems would be desirable for characterizing an image and automatically selecting the optimum processes.

REFERENCES

1. J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series", *Math. Computation*, vol. 19, Apr. 1965, pp. 297-301.
2. A. Habibi and P. A. Wintz, "Image coding by linear transformation and block quantization", *IEEE Trans. Comm. Tech.*, vol. COM-19, no. 1, Feb. 1971, pp. 50-62.
3. I. J. Good, "The interaction algorithm and practical Fourier analysis", *J. Royal Stat. Soc.*, vol. B20, 1958, pp. 361-372.
4. H. C. Andrews, *Computer Techniques in Image Processing*, New York: Academic Press, 1970.
5. E. O. Brigham, *The Fast Fourier Transform*, Prentice-Hall, Inc., 1974.
6. A. Haar, "Zur Theorie der Orthogonalen Funktionen-Systeme", *Math. Ann.*, vol. 69, 1910, pp. 331-371.
7. W. K. Pratt, W-H Chen and L. R. Welch, "Slant transform image coding", *IEEE Trans. Comm.*, vol. COM-22, no. 8, Aug. 1974, pp. 1075-1093.
8. *IEEE Trans. Audio and Electroacoustics*, June 1967, vol. AU-15 and June 1969, vol. AU-17, *Special Issues on the FFT*.
9. H. F. Harmuth, "A generalized concept of frequency and some applications", *IEEE Trans. Information Theory*, vol. IT-14, no. 3, May 1968, pp. 375-382.
10. W. K. Pratt, J. Kane and H. C. Andrews, "Hadamard transform image coding", *Proc. IEEE*, vol. 57, no. 1, Jan. 1969, pp. 58-68.
11. R. B. Blackman and J. W. Tukey, *The Measurement of Power Spectra*, New York: Dover, 1958.
12. N. Wiener, *Extrapolation, Interpolation and Smoothing of Stationary Time Series*, John Wiley, 1949.
13. W. B. Davenport and W. L. Root, *An Introduction to the Theory of Random Signals and Noise*, McGraw-Hill, 1958.

14. R. W. Schafer and L. R. Rabiner, "A digital signal processing approach to interpolation", *Proc. IEEE*, vol. 61, no. 6, June 1973, pp. 692-702.
15. T. A. Iinuma, "Image enhancement by the iterative approximation method", *Quantitative Organ Visualization in Nuclear Medicine*, ed. P. J. Kenny and E. M. Smith, University of Miami Press, 1971.
16. T. A. Iinuma and T. Nagai, "Image restoration in radioisotope imaging systems", *Phys. Med. Biol.*, vol. 12, no. 4, 1967, pp. 501-509.
17. S. M. Pizer, "Digital spatial filtering and its variations", *op. cit.* [15].
18. B. R. Hunt, "The application of constrained least squares estimation to image restoration by digital computer", *IEEE Trans. Comp.*, vol. C-22, no. 9, Sept. 1973, pp. 805-812.
19. G. F. Clemente, L. D. Marinelli and I. K. Abu-Shumays, "Regularization unfolding in low γ-activity measurements - I: evaluation of one-dimensional scanning", *International J. of Applied Radiation and Isotopes*, vol. 21, 1970, pp. 307-318.
20. D. P. MacAdam, "Digital image restoration by constrained deconvolution", *J. Opt. Soc. Am.*, vol. 60, Dec. 1970, pp. 1617-1627.
21. M. M. Sondhi, "Image restoration: the removal of spatially invariant degradations", *Proc. IEEE*, vol. 60, no. 7, July 1972, pp. 842-853.
22. W. K. Pratt, "Generalized Wiener filtering computation techniques", *IEEE Trans. Comp.*, vol. C-21, no. 7, July 1972, pp. 636-641.
23. W. H. Richardson, "Bayesian-based iterative method of image restoration", *J. Opt. Soc. Am.*, vol. 62,no. 1, Jan. 1972, pp. 55-59.
24. N. E. Hahi and T. Assefi, "Bayesian recursive image estimation", *IEEE Trans. Comp.*, vol. C-21, July 1972, pp. 734-738.

25. R. E. Kalman, "A new approach to linear filtering and prediction problems", *ASME Trans., Ser. D (J. Bas. Eng.)*, vol. 82, 1960, pp. 95-108.
26. R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory", *ASME Trans., Ser. D (J. Bas. Eng.)*, vol. 83, 1961, pp. 95-108.
27. A. Habibi, "Two-dimensional Bayesian estimate of images", *Proc. IEEE*, vol. 60, no. 7, July 1972, pp. 878-883.
28. A. K. Jain and E. Angel, "Image restoration, modelling and reduction of dimensionality", *IEEE Trans. Comp.*, vol. C23, no. 5, May 1974, pp. 470-476.
29. G. M. Robbins and T. S. Huang, "Inverse filtering for linear shift-invariant systems", *Proc. IEEE*, vol. 60, no. 7, July 1972, pp. 862-871.
30. A. A. Sawchuck, "Space-variant image motion degradation and restoration", *Proc. IEEE*, vol. 60, no. 7, July 1972, pp. 854-861.
31. A. A. Sawchuck, "Space-variant image restoration by coordinate transformation", *J. Opt. Soc. Am.*, vol. 64, no. 2, Feb. 1974, pp. 138-144.
32. R. L. Lillestrand, "Techniques for change detection", *IEEE Trans. Comp.*, vol. C-21, no. 7, July 1972, pp. 654-659.
33. A. V. Oppenheim, R. W. Schafer and T. G. Stockham, "Nonlinear filtering of multiplied and convolved signals", *Proc. IEEE*, vol. 56, Aug. 1968, pp. 1264-1291.
34. T. S. Huang, "Digital holography", *Proc. IEEE*, vol. 59, no. 9, Sept. 1971, pp. 1335-1346.
35. G. W. Stroke, "Image deblurring and aperture synthesis using *a posteriori* processing by Fourier transform holography", *Optica Acta*, vol. 16, no. 4., 1969, pp. 401-422.
36. A. vander Lugt, "A review of optical data-processing techniques", *Optica Acta*, vol. 15, no. 1, 1968, pp. 1-33.
37. W. F. Schreiber, "Picture coding", *Proc. IEEE*, vol. 55, no. 3, March 1967, pp. 320-330.

38. T. G. Stockham, "Image processing in the context of a visual model", *Proc. IEEE*, vol. 60, no. 7, July 1972, pp 828-841.
39. E. M. Lowry and J. J. De Palma, "Sine-wave response of the visual system I: the Mach phenomena", *J. Opt. Soc. Am.*, vol. 51, 1961, p 740.
40. H. C. Andrews, A. G. Tescher and R. P. Kruger, "Image processing by digital computer", *IEEE Spectrum*, July 1972.
41. H. Lipson, *Optical Transforms*, Academic Press, 1972.
42. H. C. Andrews, "Digital image restoration: a survey", *Computer*, May 1974.
43. T. S. Huang, W. F. Schreiber and O. J. Tretiak, "Image processing", *Proc. IEEE*, vol. 59, no. 11, Nov. 1971, pp. 1586-1609.
44. A. E. Todd-Pokropek, "An investigation using Monte-Carlo techniques and gaming theory into the value of digital radioistope systems", *Medical Radioisotope Scintigraphy*, vol. 1, IAEA, Vienna, 1973.
45. IAEA Co-ordinated Research Program on the Inter-comparison of Computer-assisted Scintigraphic Techniques, *op. cit.* [44].

APPENDIX A: HAAR FUNCTIONS

The Haar functions are orthogonal and complete on the interval [0,1], and are defined by the following:[6]

$$\phi_0(s) = 1 \qquad 0 \leq s \leq 1$$

$$\phi_1(s) = 1 \qquad 0 \leq s < \tfrac{1}{2}$$

$$= -1 \qquad \tfrac{1}{2} < s \leq 1$$

$$\phi_2^{(1)}(s) = \sqrt{2} \quad \text{and} \quad \phi_2^{(2)}(s) = 0 \qquad 0 \leq s < \tfrac{1}{4}$$

$$= -\sqrt{2} \qquad\qquad = 0 \qquad \tfrac{1}{4} < s < \tfrac{1}{2}$$

$$= 0 \qquad\qquad = \sqrt{2} \qquad \tfrac{1}{2} < s < \tfrac{3}{4}$$

$$= 0 \qquad\qquad = -\sqrt{2} \qquad \tfrac{3}{4} < s \leq 1$$

Dividing [0,1] into 2^n equal subintervals and denoting the j^{th} subinterval by $i_n^{(j)}$, we have that the general formula is

$$\phi_n{}^k = 0 \qquad \text{in the intervals} \qquad i_n^{(1)}, i_n^{(2)}, \ldots\ldots, i_n^{(2k-2)}$$

$$= \sqrt{(2^{n-1})} \qquad i_n^{(2k-1)}$$

$$= -\sqrt{(2^{n-1})} \qquad i_n^{(2k)}$$

$$= 0 \qquad i_n^{(2k+1)}, \ldots\ldots, i_n^{(2^n)}$$

APPENDIX B: A NOTE ON THE MICRODENSITOMETER

A microdensitometer was built to digitize images, from either transparent negatives or opaque prints, and to print them after processing. It was designed to be able to scan a picture of up to 25cm×25cm with a resolution of 0,25mm in both directions. The drum rotates at approximately 1rev/sec resulting in a sampling frequency of 1kHz.

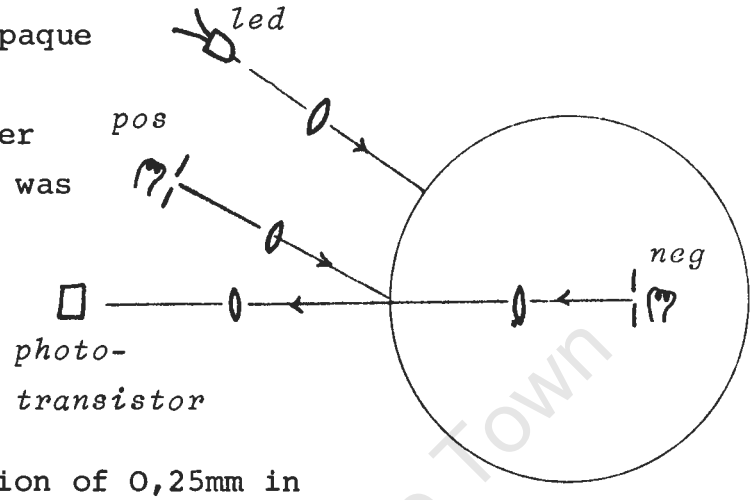
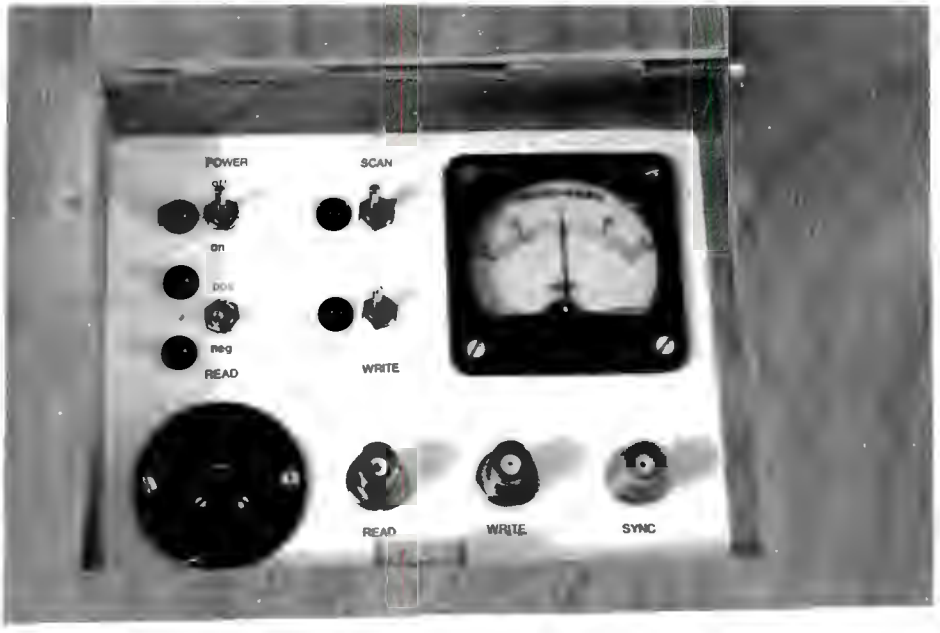


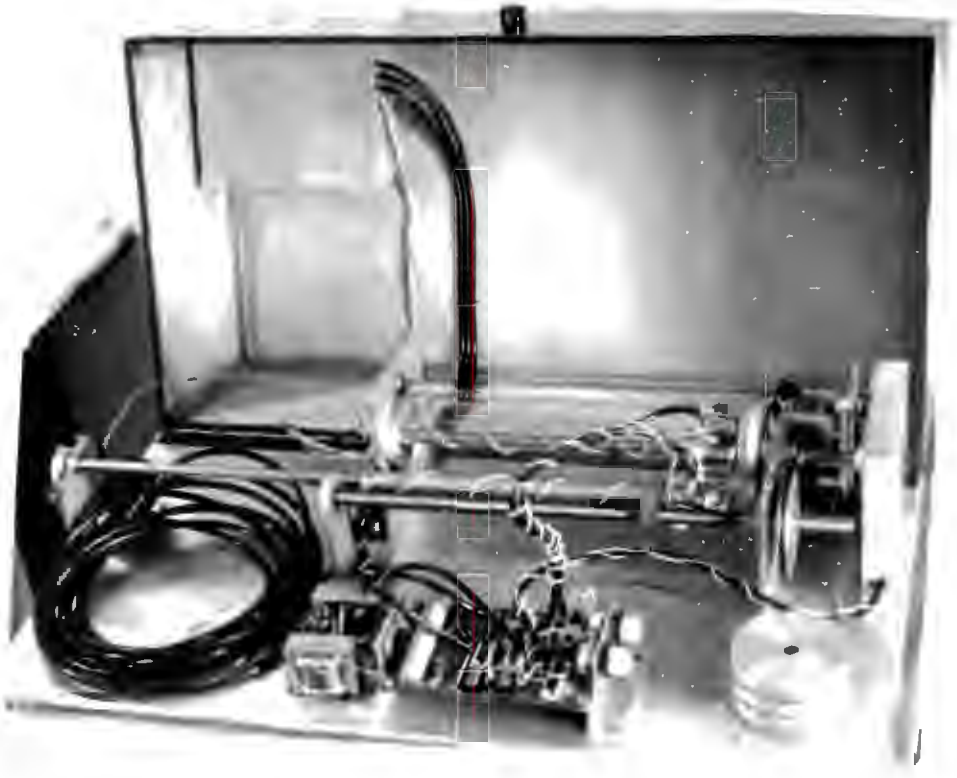
Fig. 22: Optical system in the microdensitometer.

The microdensitometer was linked to a Varian computer during operation. In read mode (digitizing an image) either the external or internal light source is switched on (see control panel shown in Pl. 32) depending on whether a print or a negative is to be scanned. A disc with zebra crossing on its circumference provides sampling interrupts for the computer. The light sources are focussed on the same point on the cylindrical drum and the rays continue by reflection or transmission to a phototransistor (Fig. 22). The amplified signal is digitized by the Varian's A/D converter and written directly on to magnetic tape. In order to print an image, a negative is wrapped around the drum and exposed during scanning by an LED. Data are output from the computer's D/A, interrupted by the same synchronizing pulses as in the scanning mode. It was found that images printed on

Plate 32:

Top: microdensitometer control panel
Bottom: inside view of microdensitometer


POWER
SCAN
on
pos
neg
READ
WRITE
READ
WRITE
SYNC



matte paper introduced least noise during image digitization. Agfapan 400 Professional 4"×5" negatives were used for printing images. A grey scale was printed and found to be adequately linear in the range 50 - 500 where 0 is ground and 2047 corresponds to +10V, the maximum output voltage of the Varian's D/A converter. A computer program was written by D. Fraser to convert Varian formatted data to Univac format so that the data could be processed in UCT's Univac computer. A program to convert the data back to Varian format was written by the author. Both are given in Appendix C.

APPENDIX C SOME COMPUTER PROGRAMS

Subroutine FOUR1 - this performs a one-dimensional Fast Fourier transform of a complex array and was written by N. Brenner of MIT Lincoln Lab.
DATA is the complex array to be transformed and comprises the real and imaginary values of the complex points in alternate locations.
N is the number of complex points, so that DATA will have 2N values.
ISIGN determines the sign of the exponent: -1 for a forward transform, +1 for the inverse.

```
      SUBROUTINE FOUR1(DATA,N,ISIGN)
      DIMENSION DATA(1024)
      IP0=2
      IP3=IP0*N
      I3REV=1
      DO 50 I3=1,IP3,IP0
      IF(I3-I3REV)10,20,20
10    TEMPR=DATA(I3)
      TEMPI=DATA(I3+1)
      DATA(I3)=DATA(I3REV)
      DATA(I3+1)=DATA(I3REV+1)
      DATA(I3REV)=TEMPR
      DATA(I3REV+1)=TEMPI
20    IP1=IP3/2
30    IF(I3REV-IP1)50,50,40
40    I3REV=I3REV-IP1
      IP1=IP1/2
      IF(IP1-IP0)50,30,30
50    I3REV=I3REV+IP1
      IP1=IP0
60    IF(IP1-IP3)70,100,100
70    IP2=IP1*2
      THETA=6.283185307/FLOAT(ISIGN*IP2/IP0)
```

```
      SINTH=SIN(THETA/2.)
      WSTPR=-2.*SINTH*SINTH
      WSTPI=SIN(THETA)
      WR=1.
      WI=0.
      DO 90 I1=1,IP1,IP0
      DO 80 I3=I1,IP3,IP2
      I2A=I3
      I2B=I2A+IP1
      TEMPR=WR*DATA(I2B)-WI*DATA(I2B+1)
      TEMPI=WR*DATA(I2B+1)+WI*DATA(I2B)
      DATA(I2B)=DATA(I2A)-TEMPR
      DATA(I2B+1)=DATA(I2A+1)-TEMPI
      DATA(I2A)=DATA(I2A)+TEMPR
80    DATA(I2A+1)=DATA(I2A+1)+TEMPI
      TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI+WR
90    WI=WI*WSTPR+TEMPR*WSTPI+WI
      IP1=IP2
      GO TO 60
100   RETURN
      END
```

Subroutine HADMD1 - this performs a one-dimensional Fast Hadamard transform.
ARRAY is the array to be transformed.
N is the number of points (real).
NORM is an option for normalizing the transform. When performing a two-dimensional transform, it is convenient to normalize only when operating in one of the two dimensions and is obviously more efficient. If NORM = 0 no normalization is performed; if NORM $\neq$ 0, then the array is divided by N.

```
      SUBROUTINE HADMD1(ARRAY,N,NORM)
      DIMENSION ARRAY(256),TEMP(256)
      M=N/2
      NN=N
      IF(N.LT.4)GO TO 60
10    DO 30 I=1,N,NN
      K=I
      MM=I+M-1
      DO 20 J=I,MM
      TEMP(J)=ARRAY(K)+ARRAY(K+1)
20    K=K+2
      K=I
      N1+MM+M
      MM=MM+I
      DO 30 J=MM,N1,2
      TEMP(J)=ARRAY(K)-ARRAY(K+1)
      TEMP(J+1)=ARRAY(K+3)-ARRAY(K+2)
30    K=K+4
      NN=NN/2
      IF(NN.LT.4)GO TO 75
      M=M/2
      DO 50 I=1,N,NN
      K=I
      MM=I+M-1
      DO 40 J=I,MM
```

```
      ARRAY(J)=TEMP(K)+TEMP(K+1)
40    K=K+2
      K=I
      N1=MM+M
      MM=MM+1
      DO 50 J=MM,N1,2
      ARRAY(J)=TEMP(K)-TEMP(K+1)
      ARRAY(J+1)=TEMP(K+3)-TEMP(K+2)
50    K=K+4
      NN=NN/2
      M=M/2
      IF(NN.GT.2)GO TO 10
60    DO 70 I=1,N
70    TEMP(I)=ARRAY(I)
75    DO 80 I=1,N,2
      ARRAY(I)=TEMP(I)+TEMP(I+1)
80    ARRAY(I+1)=TEMP(I)-TEMP(I+1)
      IF(NORM.EQ.0)RETURN
      AN=FLOAT(N)
      DO 90 I=1,N
90    ARRAY(I)=ARRAY(I)/AN
      RETURN
      END
```

Subroutine HADMD2 - this performs a two-dimensional Fast Hadamard transform and calls SUBROUTINE HADMD1.
ARRAY is the square array to be transformed.
N is the dimension of the array (N×N).
Note that a two-dimensional Haar transform may be performed by replacing calls to HADMD1 by HARFOR or HARINV for forward and inverse Haar transforms respectively.

```
      SUBROUTINE HADMD2(ARRAY,N)
      DIMENSION ARRAY(64,64)DATA(64)
      DO 30 I=1,N
      DO 10 J=1,N
10    DATA(J)=ARRAY(I,J)
      CALL HADMD1(DATA,N,0)
      DO 20 J=1,N
20    ARRAY(I,J)=DATA(J)
30    CONTINUE
      DO 60 J=1,N
      DO 40 I=1,N
40    DATA(I)=ARRAY(I,J)
      CALL HADMD1(DATA,N,1)
      DO 50 I=1,N
50    ARRAY(I,J)=DATA(I)
60    CONTINUE
      RETURN
      END
```

Subroutine HARFOR -this performs a forward Fast Haar transform. (1-D)
ARRAY is the array to be transformed.
N is the number of points in the array.

```
      SUBROUTINE HARFOR(ARRAY,N)
      DIMENSION ARRAY(256),TEMP(128)
      R2=1.41421356
      R2N=R2
      NN=N/2
      K=NN
      DO 10 I=N,1,-2
      TEMP(K)=ARRAY(I-1)+ARRAY(I)
      ARRAY(K+NN)=(ARRAY(I-1)-ARRAY(I))/R2N
10    K=K-1
      L=NN
20    K=1
      N1=L-NN/2+1
      R2N=R2N*R2
      DO 30 I=N1,L
      ARRAY(I)=(TEMP(K)-TEMP(K+1))/R2N
30    K=K+2
      IF(NN.GT.2)GO TO 40
      ARRAY(1)=(TEMP(1)+TEMP(2))/R2N
      RETURN
40    K=1
      DO 50 I=1,NN,2
      TEMP(K)=TEMP(I)+TEMP(I+1)
50    K=K+1
      NN=NN/2
      L=L-NN
      GO TO 20
      END
```

Subroutine HARINV - this performs an inverse one-dimensional Fast Haar transform.
ARRAY is the array to be transformed.
N is the number of array points.

```
      SUBROUTINE HARINV(ARRAY,N)
      DIMENSION ARRAY(256),TEMP(256)
      R2=1.41421356
      R2N=R2
      NN=N/2
      L=N
      M=NN+1
10    DO 20 I=M,L
20    ARRAY(I)=ARRAY(I)/R2N
      NN=NN/2
      IF(NN.LT.1)GO TO 30
      L=M-1
      M=M-NN
      R2N=R2N*R2
      GO TO 10
30    ARRAY(1)=ARRAY(1)/R2N
      NN=N/2
      DO 40 I=1,N
40    TEMP(I)=ARRAY(I)
      L=1
      M=1
50    K=1
      DO 60 I=1,L
      TEMP(K)=ARRAY(I)+ARRAY(I+M)
      TEMP(K+1)=ARRAY(I)-ARRAY(I+M)
60    K=K+2
      L=L+M
      M=M+M
70    K=1
      DO 80 I=1,L
      ARRAY(K)+TEMP(I)+TEMP(I+M)
      ARRAY(K+1)=TEMP(I)-TEMP(I+M)
80    K=K+2
```

```
      L=L+M
      M=M+M
      IF(M.EQ.N)RETURN
      IF(M.LT.NN)GO TO 50
      DO 90 I=1,NN
90    TEMP(I)=ARRAY(I)
      GO TO 70
      END
```

This program generates a two-dimensional Gaussian function in a 65×65 matrix. The function is written to file unit 20 in format 5X,5E14.5. The only data required is the factor of the exponent (format F5.0).

```
      DIMENSION ARRAY(65,65)
1     FORMAT(F5.0)
5     FORMAT(5X,5E14.5)
      READ(8,1)A
      DO 10 I=1,33
      DO 10 J=I,33
      L=(I-1)*(I-1)+(J-1)*(J-1)
      X=FLOAT(L)
      ARRAY(I,J)=EXP(-X/A)
10    ARRAY(J,I)=ARRAY(I,J)
      DO 20 I=1,33
      DO 20 J=1,32
      K=66-J
20    ARRAY(I,K)=ARRAY(I,J)
      DO 30 J=1,65
      DO 30 I=1,32
      K=66-I
30    ARRAY(K,J)=ARRAY(I,J)
      WRITE(20,5)((ARRAY(I,J),J=1,65),I=1,65)
      STOP
      END
```

This is the main program for performing iterative restoration in the frequency domain. It calls subroutines AMPTUD, DEVITN, FOUR1, ITERTN, MULTPY, SCALE and TRANSF.
Data required:
N - array size (64)
NTIMES - number of iterations to be performed
NOPT - 0 if the image array is real, 1 if imaginary
NREMEM - the iteration images (in order) which are required to be written to file. The i^{th} image is written to file number i+15
R - restoring filter function (complex)
SMOOTH - smoothing function (complex)
G - image to be restored

```
      DIMENSION GO(64,128),NREMEM(15),R(64,128),G(64,128)
      DIMENSION GN(64,128),SMOOTH(64,128),STORE(64,128)
5     FORMAT('1',T42,'IMAGE RESTORATION BY FREQUENCY DOMAIN ITERATION'/T
     142,47('*')//)
10    FORMAT(18I2)
20    FORMAT(2(E26.5,E14.5))
30    FORMAT(5X,5E14.5)
40    FORMAT('ARRAY ',5E14.5)
      READ( 8,10)N,NTIMES,NOPT,(NREMEM(I),I=1,NTIMES)
      NN=2*N
      READ(8,20)((R(I,J),J=1,NN),I=1,N)
      READ(8,20)((SMOOTH(I,J),J=1,NN),I=1,N)
      IF(NOPT.EQ.1)GO TO 200
      READ(8,30)((G(I,J),J=1,N),X,I=1,N)
      M=N+1
      READ(8,30)(X,I=1,M)
      CALL TRANSF(G,N,-1,NOPT)
      CALL MULTPY(G,SMOOTH,N)
      GO TO 250
200   READ(8,20)((G(I,J),J=1,NN),I=1,N)
250   CALL SCALE(G,N,NN,50.)
      DO 280 I=1,N
      DO 280 J=1,NN
      GN(I,J)=G(I,J)
280   STORE(I,J)=G(I,J)
```

```
      WRITE(5,5)
      CALL TRANSF(STORE,N,1,1)
      CALL AMPTUD(STORE,GO,N)
      SUM=0.
      DO 285 I=1,N
      DO 285 J=1,N
      SUM=SUM+GO(I,J)
      SMOOTH(I,J+N)=GO(I,J)
285   GO(I,J)=SQRT(GO(I,J))
      DO 310 K=1,NTIMES
      L=K+15
      CALL ITERTN(G,GN,N,R)
      DO 290 I=1,N
      DO 290 J=1,NN
290   STORE(I,J)=GN(I,J)
      CALL TRANSF(STORE,N,1,1)
      CALL AMPTUD(STORE,SMOOTH,N)
      IF(NREMEM(1).NE.K)GO TO 305
      DO 293 I=2,NTIMES
293   NREMEM(I-1)=NREMEM(I)
      DO 300 I=1,N
      DO 300 J=1,N
300   STORE(I,J)=SMOOTH(I,J)
      CALL SCALE(STORE,N,N,40.)
      WRITE(L,40)((STORE(I,J),J=1,N),STORE(I,1),I=1,N)
      WRITE(L,40)(STORE(1,J),J=1,N),STORE(1,1)
305   CALL DEVITN(SMOOTH,N,GO,SUM)
310   CONTINUE
      STOP
      END
```

Subroutine AMPTUD - determines the real amplitude of a complex array.

```
      SUBROUTINE AMPTUD(ARRAY1,ARRAY2,N)
      DIMENSION ARRAY1(64,128),ARRAY2(64,128)
      DO 100 I=1,N
      DO 100 J=1,N
      K=2*J
100   ARRAY2(I,J)=SQRT(ARRAY1(I,K-1)**2+ARRAY1(I,K)**2)
      RETURN
      END
```

Subroutine DEVITN - determines the convergence of the iteration images.

```
      SUBROUTINE DEVITN(SMOOTH,N,GO,SUM)
      DIMENSION SMOOTH(64,128),GO(64,128)
10    FORMAT(' NORM SQUARE DIFF =',F16.5,';  MEAN DIFF IN STD DEV =',
     &F12.5,';  TOTAL =',F16.2,';  MAX ELT =',F16.2)
      SUMA=0.
      BIG=0.
      TOTAL=0.
      SUM1=0.
      DO 100 I=1,N
      DO 100 J=1,N
      L=J+N
      A=ABS(SMOOTH(I,J)-SMOOTH(I,L))
      TOTAL=TOTAL+SMOOTH(I,J)
      IF(BIG.LT.SMOOTH(I,J))BIG=SMOOTH(I,J)
      SUMA=SUMA+A*A
      IF(GO(I,J).LT.0.001)GO TO 50
      SUM1+SUM1+A/GO(I,J)
50    SMOOTH(I,L)=SMOOTH(I,J)
100   CONTINUE
      SUMA=SUMA/SUM
      SUM1=SUM1/4096.
      WRITE(5,10)SUMA,SUM1,TOTAL,BIG
      RETURN
      END
```

Subroutine ITERTN - calculates the next iteration.

```
      SUBROUTINE ITERTN(G,GN,N,R)
      DIMENSION R(64,128),G(64,128),GN(64,128)
      NN=2*N
      CALL MULTPY(GN,R,N)
      DO 100 I=1,64
      DO 100 J=1,128
100   GN(I,J)=GN(I,J)+G(I,J)
      RETURN
      END
```

Subroutine MULTPY - multiplies two square complex matrices

```
      SUBROUTINE MULTPY(ARRAY1,ARRAY2,N)
      DIMENSION ARRAY1(64,128),ARRAY2(64,128)
      M=2*N-1
      DO 100 I=1,N
      DO 100 J=1,M,2
      TEMP=ARRAY1(I,J)
      ARRAY1(I,J)=ARRAY1(I,J)*ARRAY2(I,J)-ARRAY1(I,J+1)*ARRAY2(I,J+1)
100   ARRAY1(I,J+1)=TEMP*ARRAY2(I,J+1)+ARRAY1(I,J+1)*ARRAY2(I,J)
      RETURN
      END
```

Subroutine SCALE - scales a matrix to a max. value.

```
      SUBROUTINE SCALE(ARRAY,M,N,AMAX)
      DIMENSION ARRAY(64,128)
      BIG=0.
      DO 100 I=1,M
      DO 100 J=1,N
100   IF(BIG.LT.ABS(ARRAY(I,J)))BIG=ABS(ARRAY(I,J))
      BIG=AMAX/BIG
      DO 200 I=1,M
      DO 200 J=1,N
200   ARRAY(I,J)=ARRAY(I,J)*BIG
      END
```

Subroutine TRANSF - performs a two-dimensional FFT.

```
      SUBROUTINE TRANSF(ARRAY,N,ISIGN,NOPT)
      DIMENSION ARRAY(64,128),DATA(128)
      NN=2*N
      IF(NOPT.EQ.1)GO TO 200
      DO 100 I=1,N
      DO 100 J=N,1,-1
      K=2*J
      ARRAY(I,K-1)=ARRAY(I,J)
100   ARRAY(I,K)=0.
200   DO 250 I=1,N
      DO 230 J=1,NN
230   DATA(J)=ARRAY(I,J)
      CALL FOUR1(DATA,N,ISIGN)
      DO 240 J=1,NN
240   ARRAY(I,J)=DATA(J)
250   CONTINUE
      M=NN-1
      DO 350 J=1,M,2
      DO 330 I=1,N
      K=2*I
      DATA(K-1)=ARRAY(I,J)
330   DATA(K)=ARRAY(I,J+1)
      CALL FOUR1(DATA,N,ISIGN)
      DO 340 I=1,N
      K=2*I
      ARRAY(I,J)=DATA(K-1)
340   ARRAY(I,J+1)=DATA(K)
350   CONTINUE
      RETURN
      END
```

This program converts data in PDP format to Univac format.

```
      DIMENSION IDATA(2740),JDATA(4098),ARRAY(65,65)
10    FORMAT('1',T50,'CONTENTS OF TAPE BLOCK NUMBER',I3/T50,32('*'))
20    FORMAT(/' LINE ',I2,':')
30    FORMAT(16I7)
40    FORMAT('ARAY ',5E14.5)
50    FORMAT(I2)
      READ(8,50)NFLS
      CALL NTRAN(19,8,NFLS,22)
      CALL NTRAN(19,2,2740,IDATA,L,22)
      J=1
      DO 90 I=1,2731,2
      JDATA(J)=FLD(12,12,IDATA(I))
      JDATA(J+1)=FLD(0,12,IDATA(I+1))
      JDATA(J+2)=FLD(24,12,IDATA(I+1))
90    J=J+3
      JDATA(1)=FLD(0,12,IDATA(1))
      DO 100 I=1,8
      WRITE(5,10)JDATA(1)
      DO 100 J=1,8
      K=8*(I-1)+J-1
      M=K*64+2
      N=M+63
      WRITE(5,20)K
100   WRITE(5,30)(JDATA(L),L=M,N)
      DO 110 I=1,64
      K=65-I
      DO 110 J=1,64
      L=65-J
      M=64*(I-1)+J+1
110   ARRAY(L,K)=FLOAT(JDATA(M))
      DO 120 I=1,64
      ARRAY(65,I)=ARRAY(64,I)
120   ARRAY(I,65)=ARRAY(I,64)
      ARRAY(65,65)=ARRAY(64,64)
      WRITE(20,40)((ARRAY(I,J),J=1,65),I=1,65)
      STOP
      END
```

This program converts Varian formatted data to Univac format, and is a corrected version of one written by D. Fraser.
Data required:
NEOF - number of EOF's to be skipped on tape
NSKIP - number of blocks to be skipped
NSIZE - block size (256)
NBLOK - number of blocks of size NSIZE. (256 Univac words or 576 Varian words)
LOC1 - associated variable for DA file (1).
12 is the logical number for the tape file; 13 for the DA disc file.

```
      DIMENSION IBUF(256),NDATA(576),IDATA(2048)
      DEFINE FILE 13(100,2048,U,LOC1)
      ITAPE=12
      IRD=3
      ISKP=7
      IEOF=8
      IRW=10
      IEND=22
      I1=COMPL(0)
      I2=FLD(20,16,I1)
      I3=XOR(I1,I2)
      READ(8,1000)NEOF,NSKIP,NSIZE,NBLOK,LOC1
1000  FORMAT()
      CALL NTRAN(ITAPE,IEOF,NEOF,IEND)
      CALL NTRAN(ITAPE,ISKP,NSKP,IEND)
      NW=0
      N4=NSIZE/4
      NTOT=N4*9
      N=-NTOT
      DO 100 I=1,NBLOK
      CALL NTRAN(ITAPE,IRD,NSIZE,IBUF,ITST,IEND)
      IF(ITST.LT.0)GO TO 50
      IF(ITST.NE.NSIZE)WRITE(5,1005)ITST
1005  FORMAT(/' WORDS READ < 256, ='I4/)
      DO 70 J=1,N4
      K=9*(J-1)
      L=4*(J-1)
```

```
      NDATA(K+1)=FLD(0,16,IBUF(L+1))
      NDATA(K+2)=FLD(16,16,IBUF(L+1))
      NDATA(K+3)=4096*FLD(32,4,IBUF(L+1))+FLD(0,12,IBUF(L+2))
      NDATA(K+4)=FLD(12,16,IBUF(L+2))
      NDATA(K+5)=256*FLD(28,8,IBUF(L+2))+FLD(0,8,IBUF(L+3))
      NDATA(K+6)=FLD(8,16,IBUF(L+3))
      NDATA(K+7)=16*FLD(24,12,IBUF(L+3))+FLD(0,4,IBUF(L+4))
      NDATA(K+8)=FLD(4,16,IBUF(L+4))
      NDATA(K+9)=FLD(20,16,IBUF(L+4))
70    CONTINUE
      N=N+NTOT
      WRITE(5,1070)(NDATA(J),J=1,16)
1070  FORMAT(16I6)
      DO 90 J=1,NTOT
      IF(FLD(20,1,NDATA(J)).EQ.1)NDATA(J)=OR(NDATA(J),I3)-1
      K=J+N
      IDATA(K)=NDATA(J)
      IF(K.LT.2048)GO TO 90
      N=N-2048
      NW=NW+1
      WRITE(13'LOC1,ERR=200)IDATA
      IF(NW.LE.2)WRITE(5,1050)(IDATA(L),L=1,2048)
1050  FORMAT(24I5)
      IF(NW.EQ.100)GO TO 300
90    CONTINUE
100   CONTINUE
      GO TO 300
50    IG=-ITST
      GO TO (10,20,30,40),IG
10    WRITE(5,1010)I
1010  FORMAT(/' TRANS INCOMPL READING BLOCK 'I4/)
      GO TO 300
20    WRITE(5,1020)I
1020  FORMAT(/' EOF FOUND READING BLOCK 'I4/)
      GO TO 300
30    WRITE(5,1030)I
1030  FORMAT(/' DEVICE ERROR READING BLOCK 'I4/)
      GO TO 300
```

```
40    WRITE(5,1040)I
1040  FORMAT(/' TRANS ABORTED, PRREV ERROR, READING BLOCK 'I4/)
      GO TO 300
200   J=INSTAT(J)
      WRITE(5,1100)J,NW
1100  FORMAT(/' ERROR 'O12,' IN DISC WRITE 'I4/)
300   WRITE(5,1060)NW
1060  FORMAT(/I5,' BLOCKS OF 2048 WRITTEN TO 13'/)
      CALL NTRAN(ITAPE,IRW,IEND)
      END
```

This program converts Univac formatted data to Varian format. Data required is the same as that for the previous program.

```
      DIMENSION IBUF(256),NDATA(576),IDATA(2048)
      DEFINE FILE 13(100,2048,U,LOC1)
      ITAPE=12
      IWT=1
      ISKP=7
      IEOF=8
      IRW=10
      IEND=22
      READ(8,1000)NEOF,NSKIP,NSIZE,NBLOK,LOC1
 1000 FORMAT()
      CALL NTRAN(ITAPE,IEOF,NEOF,IEND)
      CALL NTRAN(ITAPE,ISKP,NSKP,IEND)
      N4=NSIZE/4
      NTOT=N4*9
      N=-NTOT
      NW=1
      READ(13'LOC1,ERR=200)IDATA
      DO 100 I=1,NBLOK
      N=N+NTOT
      DO 60 J=1,NTOT
      K=J+N
      NDATA(J)=IDATA(K)
      IF(NDATA(J).LT.0)NDATA(J)=NDATA(J)+1
      IF(K.LT.2048)GO TO 60
      N=N-2048
      NW=NW+1
      READ(13'LOC1,ERR=200)IDATA
   60 CONTINUE
      DO 70 J=1,N4
      K=9*(J-1)
      L=4*(J-1)
      FLD(0,16,IBUF(L+1))=FLD(20,16,NDATA(K+1))
      FLD(16,16,IBUF(L+1))=FLD(20,16,NDATA(K+2))
      FLD(32,4,IBUF(L+1))=FLD(20,4,NDATA(K+3))
      FLD(0,12,IBUF(L+2))=FLD(24,12,NDATA(K+3))
      FLD(12,16,IDATA(L+2))=FLD(20,16,NDATA(K+4))
```

```
      FLD(28,8,IDATA(L+2))=FLD(20,8,NDATA(K+5))
      FLD(0,8,IBUF(L+3))=FLD(28,8,NDATA(K+5))
      FLD(8,16,IBUF(L+3))=FLD(20,16,NDATA(K+6))
      FLD(24,12,IBUF(L+3))=FLD(20,12,NDATA(K+7))
      FLD(0,4,IBUF(L+4))=FLD(32,4,NDATA(K+7))
      FLD(4,16,IBUF(L+4))=FLD(20,16,NDATA(K+8))
      FLD(20,16,IBUF(L+4))=FLD(20,16,NDATA(K+9))
70    CONTINUE
      WRITE(5,1001)(NDATA(K),K=1,18)
      WRITE(5,1002)(IBUF(K),K=1,8)
1001  FORMAT(18I7)
1002  FORMAT(8(2X,O12))
      CALL NTRAN(ITAPE,IWT,NSIZE,IBUF,ITST,IEND)
      IF(ITST.LT.0)GO TO 50
      IF(ITST.NE.NSIZE)WRITE(5,1005)ITST
1005  FORMAT(/' WORDS WRITTEN < 256, ='I4/)
100   CONTINUE
      GO TO 300
50    IG=-ITST
      GO TO (10,20,30,40),IG
10    WRITE(5,1010)I
1010  FORMAT(/' TRANS INCOMPL WRITING BLOCK 'I4/)
      GO TO 300
20    WRITE(5,1020)I
1020  FORMAT(/' EOF FOUND WRITING BLOCK 'I4/)
      GO TO 300
30    WRITE(5,1030)I
1030  FORMAT(/' DEVICE ERROR WRITING BLOCK 'I4/)
      GO TO 300
40    WRITE(5,1040)I
1040  FORMAT(/' TRANS ABORTED, PRREV ERROR, WRITING BLOCK 'I4/)
      GO TO 300
200   J=INSTAT(J)
      WRITE(5,1100)J,NW
1100  FORMAT(/' ERROR 'O12,' IN DISC READ 'I4/)
300   WRITE(5,1060)NW
```

```
1060 FORMAT(/I5,' BLOCKS OF 2048 READ FROM 13'/)
     WRITE(5,1070)I
1070 FORMAT(/I5,' BLOCKS OF 576 WRITTEN TO TAPE')
     CALL NTRAN(ITAPE,IRW,IEND)
     STOP
     END
```

This program restores an image by performing density shifting.
It call subroutines AMPTUD, MULTPY, SCALE, FOUR1, and TRANSF.
Data required:
N - array size (64)
N1 - size of approximate PSR function (15)
PICIN - image to be restored
RSPONS - PSR function
SMOOTH - complex smoothing function.

```
      DIMENSION PICIN(64,64),PICOUT(64,128),RSPONS(31,31)
      DIMENSION SMOOTH(64,128)
   10 FORMAT(2I2)
   20 FORMAT(5X,5E14.5)
   30 FORMAT('ARAY ',5E14.5)
   40 FORMAT(2(E26.5,E14.5))
      READ(8,10)N,N1
      NN=2*N
      M=N+1
      AN=FLOAT(N)
      READ(8,20)((PICIN(I,J),J=1,N),X,I=1,N)
      READ(8,20)(X,I=1,M)
      READ(8,20)((RSPONS(I,J),J=1,N1),I=1,N1)
      READ(8,40)((SMOOTH(I,J),J=1,NN),I=1,N)
      NDIF=(N1-1)/2
      DO 100 I=1,N
      DO 100 J=1,N
  100 PICOUT(I,J)=0.
      DO 200 I=1,N
      I1=I-NDIF
      I2=I+NDIF
      DO 200 J=1,N
      J1=J-NDIF
      J2=J+NDIF
      SUMK1=0.
      SUMK2=0.
```

```
      SUML1=0.
      SUML2=0.
      IN=0
      DO 150 K=I1,I2
      IN=IN+1
      JN=0
      KK=K
      IF(K.LT.1)KK=1
      IF(K.GT.N)KK=N
      DO 150 L=J1,J2
      JN=JN+1
      LL=L
      IF(L.LT.1)LL=1
      IF(L.GT.N)LL=N
      TEMP=PICIN(KK,LL)*RSPONS(IN,JN)
      SUMK1=SUMK1+TEMP
      SUMK2=SUMK2+TEMP*FLOAT(K)
      SUML1=SUML1+TEMP
  150 SUML2=SUML2+TEMP*FLOAT(L)
      IF(ABS(SUMK1).LT.0.001)GO TO 160
      COORDX=SUMK2/SUMK1+.5
      INEW=INT(COORDX)
      IF(COORDX.LT.1.)INEW=1
      IF(COORDX.GT.AN)INEW=N
      GO TO 170
  160 INEW=I
  170 IF(ABS(SUML1).LT.0.001)GO TO 180
      COORDY=SUML2/SUML1+.5
      JNEW=INT(COORDY)
      IF(COORDY.LT.1.)JNEW=1
      IF(COORDY.GT.AN)JNEW=N
      GO TO 190
  180 JNEW=J
  190 PICOUT(INEW,JNEW)=PICOUT(INEW,JNEW)+PICIN(I,J)
  200 CONTINUE
      CALL SCALE(PICOUT,N,N,40.)
      WRITE(19,30)((PICOUT(I,J),J=1,N),PICOUT(I,N),I=1,N)
      WRITE(19,30)(PICOUT(N,I),I=1,N),PICOUT(N,N)
      CALL TRANSF(PICOUT,N,-1,0)
```

```
CALL MULTPY(PICOUT,SMOOTH,N)
CALL TRANSF(PICOUT,N,1,1)
CALL AMPTUD(PICOUT,PICIN,N)
CALL SCALE(PICIN,N,N,40.)
WRITE(20,30)((PICIN(I,J),J=1,N),PICIN(I,1),I=1,N)
WRITE(20,30)(PICIN(1,J),J=1,N),PICIN(1,1)
STOP
END
```

This program modifies the density histogram of an image. In the version presented here, the equalizing function is determined and stored in array AR. If a different type of modification is required, then AR should be suitably defined.

Data required:

ARRAY - the image array to be modified

```
      DIMENSION ARRAY(64,64),INT(460),IND(460),AR(460)
      DIMENSION IBUF(1000),TPLOT(460)
5     FORMAT('1')
10    FORMAT(5X,5E14.5)
20    FORMAT(10I8)
      READ(8,10)((ARRAY(I,J),J=1,64),X,I=1,64)
      BIG=0.
      DO 100 I=1,64
      DO 100 J=1,64
100   IF(BIG.LT.ARRAY(I,J))BIG=ARRAY(I,J)
      BIG=450./BIG
      DO 110 I=1,64
      DO 110 J=1,64
      ARRAY(I,J)=ARRAY(I,J)*BIG
      NUM=IFIX(ARRAY(I,J))+1
110   IND(NUM)=IND(NUM)+1
      INT(1)=0
      IND(450)=IND(450)+IND(451)
      DO 120 I=9,450
120   INT(I+1)=INT(I)+IND(I)
      BIG=442./FLOAT(INT(451))
      DO 130 I=10,451
130   AR(I)=FLOAT(INT(I))*BIG+8.
      X=0.
      DO 140 I=1,9
      AR(I)=X
140   X=X+1.
      AR(452)=451.
      AR(453)=452.
```

```
      WRITE(5,5)
      WRITE(5,20)(IND(I),I=1,451)
      CALL PLOTS(IBUF,1000,7)
      CALL PLOT(5.,5.,-3)
      CALL PLOT(12.,0.,2)
      CALL PLOT(12.,16.,2)
      CALL PLOT(0.,16.,2)
      CALL PLOT(0.,0.,2)
      CALL PLOT(3.,12.,-3)
      CALL PLOT(6.,0.,2)
      BIG=0.
      DO 143 I=1,451
      TPLOT(I)=FLOAT(IND(I))
  143 IF(BIG.LT.TPLOT(I))BIG=TPLOT(I)
      BIG=2./BIG
      DO 144 I=1,451
  144 TPLOT(I)=TPLOT(I)*BIG
      DEL=8./450.
      CALL PLOT(0.,TPLOT(1),3)
      X=DEL
      DO 145 I=2,451
      CALL PLOT(X,TPLOT(I),2)
  145 X=X+DEL
      CALL PLOT(0.,-1.5,-3)
      CALL PLOT(6.,0.,2)
      BIG=0.
      DO 146 I=25,451
  146 IF(BIG.LT.TPLOT(I))BIG=TPLOT(I)
      DO 147 I=25,451
  147 TPLOT(I)=TPLOT(I)/BIG
      X=DEL*24.
      CALL PLOT(X,TPLOT(25),3)
      DO 148 I=26,451
      X=X+DEL
  148 CALL PLOT(X,TPLOT(I),2)
      CALL EQUAL
      DO 150 I=1,452
  150 IND(I)=0.
      DO 160 I=1,64
```

```
      DO 160 J=1,64
      NUM=IFIX(ARRAY(I,J))+1
160   IND(NUM)=IND(NUM)+1
      IND(450)=IND(450)+IND(451)
      WRITE(5,5)
      WRITE(5,20)( IND(I),I=1,451)
      WRITE(20,10)((ARRAY(I,J),J=1,64),x,I=1,64)
      BIG=0.
      DO 170 I=1,451
      TPLOT(I)=FLOAT(IND(I))
170   IF(BIG.LT.TPLOT(I))BIG=TPLOT(I)
      BIG=2./BIG
      DO 171 I=1,451
171   TPLOT(I)=TPLOT(I)*BIG
      X=DEL
      CALL PLOT(0.,-5.,-3)
      CALL PLOT(6.,0.,2)
      CALL PLOT(0.,TPLOT(1),3)
      DO 172 I=2,451
      CALL PLOT(X,TPLOT(I),2)
172   X=X+DEL
      CALL PLOT(0.,-1.5,-3)
      CALL PLOT(6.,0.,2)
      BIG=0.
      DO 174 I=25,451
174   IF(BIG.LT.TPLOT(I))BIG=TPLOT(I)
      DO 175 I=25,451
175   TPLOT(I)=TPLOT(I)/BIG
      X=DEL*24.
      CALL PLOT(X,TPLOT(1),3)
      DO 176 I=26,451
      X=X+DEL
176   CALL PLOT(X,TPLOT(I),2)
      CALL PLOT(0.,4.,-3)
      CALL PLOT(6.,0.,2)
      X=DEL
      CALL PLOT(0.,0.,3)
```

```
      DO 173 I=2,451
      A=AR(I)/225.
      CALL PLOT(X,A,2)
173   X=X+DEL
      CALL PLOT(20.,-20.,999)
      STOP
      SUBROUTINE EQUAL
      DO 300 I=1,64
      DO 300 J=1,64
      IF(ARRAY(I,J).LT.8.)GO TO 300
      N=IFIX(ARRAY(I,J))
      A=ARRAY(I,J)-FLOAT(N)
      N=N+2
      F1=(AR(N)+AR(N-1))/2.
      F2=AR(N)-AR(N-1)
      F3=(AR(N+1)-AR(N)-AR(N-1)+AR(N-2))/4.
      F4=(AR(N+1)-3.*AR(N)+3.*AR(N-1)-AR(N-2))/6.
300   ARRAY(I,J)=F1+(A-.5)*F2+A*(A-1.)*F3+A*(A-.5)*(A-1.)*F4
      RETURN
      END
```

This program interpolates a 64×64 matrix to form a 256×256 matrix for display on the microdensitometer. The first 4 terms of Bessel's central difference interpolation formula are used.

Data required:
ARRAY - 64×64 array to be interpolated

```
      DIMENSION ARRAY(256,64),IDATA(2048),TEMP(256),T(64)
      DEFINE FILE 13(100,2048,U,LOC1)
10    FORMAT(5X,5E14.5)
      READ(8,10)((ARRAY(I,J),J=1,64),I=1,64)
      WRITE(13'LOC1,ERR=200)IDATA
      BIG=0.
      DO 20 I=1,64
      DO 20 J=1,64
20    IF(BIG.LT.ARRAY(I,J))BIG=ARRAY(I,J)
      BIG=450./BIG
      DO 30 I=1,64
      DO 30 J=1,64
30    ARRAY(I,J)=BIG*ARRAY(I,J)
      DO 40 J=1,64
      DO 35 I=1,64
35    T(I)=ARRAY(I,J)
      CALL EXPAN
      DO 40 I=1,256
40    ARRAY(I,J)=TEMP(I)
      DO 70 I=0,255,8
      M=-256
      DO 60 K=1,8
      L=I+K
      M=M+256
      DO 45 J=1,64
45    T(J)=ARRAY(L,J)
      CALL EXPAN
      DO 50 J=1,255
```

```
50    IDATA(J+M)=IFIX(TEMP(J))+50.
60    IDATA(256+M)=0
      WRITE(13'LOC1,ERR=200)IDATA
70    CONTINUE
      DO 80 I=1,2048
80    IDATA(I)=0
      WRITE(13'LOC1,ERR=200)IDATA
      STOP
200   WRITE(5,1)
1     FORMAT(' ERROR')
      STOP
      SUBROUTINE EXPAN
      TEMP(1)=T(1)
      TT=T(2)-T(1)
      TEMP(2)=T(1)+TT/4.
      TEMP(3)=T(1)+TT/2.
      TEMP(4)=T(1)+3.*TT/4.
      TEMP(5)=T(2)
      NC=5
      DO 100 N=3,63
      F1=(T(N)+T(N-1))/2.
      F2=T(N)-T(N-1)
      F3=(T(N+1)-T(N)-T(N-1)+T(N-2))/4.
      F4=(T(N+1)-3.*T(N)+3.*T(N-1)-T(N-2))/6.
      DEL=.25
      DO 95 N1=1,3
      TEMP(N1+NC)=F1+(DEL-.5)*F2+DEL*(DEL-1.)*F3
      TEMP(N1+NC)=TEMP(N1+NC)+DEL*(DEL-.5)*(DEL-1.)*F4
95    DEL=DEL+.25
      NC=NC+4
100   TEMP(NC)=T(N)
      TT=T(63)-T(64)
      TEMP(250)=T(63)-TT/4.
      TEMP(251)=T(63)-TT/2.
      TEMP(252)=T(63)-3.*TT/4.
      TEMP(253)=T(64)
      RETURN
      END
```

This program plots a binary Lee hologram from a complex array.
Data required:
NTIME - maximum plotting time in minutes
FACT - scale factor for plotting (unity gives 25" square plot)
ARRAY - complex array.

```
      DIMENSION ARRAY(65,130),IBUF(1000)
5     FORMAT(I3,F4.0)
10    FORMAT(2(E26.5,E14.5))
20    FORMAT(' J=',I4)
      READ(8,5)NTIME,FACT
      N=65
      M=2*N
      N1=N
      N2=1
      N3=-1
      AN=FLOAT(N)
      FACT=FACT*25./AN
      READ(8,10)((ARRAY(I,J),J=1,M),I=1,N)
      CALL SCALE(ARRAY,N,M,4.)
      CALL PLOTS(IBUF,1000,7)
      CALL PLTIME(NTIME)
      CALL FACTOR(FACT)
      DIST=4.+AN
      CALL PLOT(DIST,0.,2)
      CALL PLOT(DIST,DIST,2)
      CALL PLOT(0.,DIST,2)
      CALL PLOT(0.,0.,2)
      CALL PLOT(2.5,2.5,-3)
      DO 300 J=1,N
      K=2*J
      K1=K-1
      DO 100 I=N1,N2,N3
      AREAL=ARRAY(I,K1)
      AIMAG=ARRAY(I,K)
      NN=N-I+1
100   CALL RECT(AREAL,AIMAG,NN,J)
```

```
      N1=N-N1+1
      N2=N-N2+1
      N3=-N3
      WRITE(5,20)J
  200 CONTINUE
      CALL PLOT(10.,-2.,999)
      STOP
      END


      SUBROUTINE RECT(AR,AI,I,J)
      X=FLOAT(J-1)
      Y=FLOAT(I-1)
      IF(ABS(AR).LT.0.00001)GO TO 20
      IF(AR.LT.0.)GO TO 10
      YY=Y+.375
      CALL RECPLT(X,YY,AR)
      GO TO 20
   10 YY=Y-.125
      AR=ABS(AR)
      CALL RECPLT(X,YY,AR)
   20 IF(ABS(AI).LT.0.00001)RETURN
      IF(AI.LT.0.)GO TO 30
      YY=Y+.125
      CALL RECPLT(X,YY,AI)
      RETURN
   30 AI=ABS(AI)
      YY=Y-.375
      CALL RECPLT(X,YY,AI)
      RETURN
      END
```

```
      SUBROUTINE RECPLT(X,Y,A)
      IF(A.LT.1.)GO TO 10
      A=A/8.
      DEL=.125
      GO TO 20
10    A=SQRT(A)/8.
      DEL=A
20    XMIN=X-A
      XMAX=X+A
      YMIN=Y-DEL
      YMAX=Y+DEL
      CALL PLOT(XMIN,YMIN,3)
      CALL PLOT(XMAX,YMIN,2)
      CALL PLOT(XMAX,YMAX,2)
      CALL PLOT(XMIN,YMAX,2)
      CALL PLOT(XMIN,YMIN,2)
      RETURN
      END
```

This program produces an isometric display of a 65×65 distribution.

Data required:

NOPT - if 0 then array displayed in given form; if 1 then the array is shifted half a period, which is used for displaying DFT's in more familiar form.

NLINE - if 0 then lines are drawn in both dimensions; if 1 then only horizontal lines are drawn

AMAX - maximum displacement of horizontal lines (typically 1.0 to 2.0)

CHAR - a title of up to 48 characters

ARRAY - the array to be displayed

```
      DIMENSION ARRAY(65,65),HMAX(131),NPEN(65,65),IBUF(5000)
      DIMENSION CHAR(8),TEMP(65,65),HTEMP(130),HTEMP1(130)
      DIMENSION ROW(2,100),IPEN(100)
10    FORMAT(2I1,3F5.0)
15    FORMAT(8A6)
20    FORMAT(5X,5E14.5)
25    FORMAT(' ','ONLY HORIZONTAL LINES TO BE PLOTTED')
      HTEMP(130)=0.
      HTEMP1(130)=0.
      READ(8,10)NOPT,NLINE,AMAX
      IF(NLINE.EQ.1)WRITE(5,25)
      READ(8,15)(CHAR(I,),I=1,8)
      GRIDX=.125
      IF(NOPT.EQ.1)GO TO 30
      READ(8,20)((ARRAY(I,J),J=1,65),I=1,65)
      GO TO 60
30    READ(8,20)((TEMP(I,J),J=1,65),I=1,65)
      DO 40 I=1,32
      DO 40 J=1,32
      ARRAY(I,J)=TEMP(I+32,J+32)
      ARRAY(I+32,J)=TEMP(I,J+32)
      ARRAY(I,J+32)=TEMP(I+32,J)
40    ARRAY(I+32,J+32)=TEMP(I,J)
      DO  50 I=1,64
      ARRAY(65,I)=ARRAY(1,I)
      ARRAY(I,65)=ARRAY(I,1)
```

```
50    CONTINUE
      ARRAY(65,65)=ARRAY(1,1)
60    GRIDY=GRIDX*0.86603
      BIG=0.
      DO 70 I=1,65
      DO 70 J=1,65
70    IF(BIG.LT.ARRAY(I,J))BIG=ARRAY(I,J)
      BIG=AMAX/BIG
      DO 80 I=1,65
      DELTAY=FLOAT(65-I)*GRIDY
      DO 80 J=1,65
80    ARRAY(I,J)=ARRAY(I,J)*BIG+DELTAY
      DO 90 J=1,65
      M=2*J-1
90    HMAX(M)=ARRAY(65,J)
      DO 91 J=2,128,2
91    HMAX(J)=(HMAX(J+1)+HMAX(J-1))/2.
      DO 92 J=1,65
92    NPEN(65,J)=2
      DO 140 I=64,1,-1
      L=I+1
      DO 95 J=1,129
95    HTEMP(J)=HMAX(J)
      DO 120 J=1,64
      K=2*J
      IF(ARRAY(I,J).GE.HMAX(K))GO TO 100
      NPEN(I,J)=3
      HMAX(K-1)=HMAX(K)
      GO TO 110
100   NPEN(I,J)=2
      HMAX(K-1)=ARRAY(I,J)
110   HMAX(K)=HMAX(K+1)
120   CONTINUE
      NPEN(I,65)=2
      HMAX(129)=ARRAY(I,65)
      DO 130 J=2,128,2
      H=(HMAX(J-1)+HMAX(J+1))/2.
130   IF(HMAX(J).LT.H)HMAX(J)=H
```

```
      DO 1500 J=1,65
      TEMP(I,J)=0.
      FRAC1=0.
      FRAC2=0.
      IF(NPEN(I,J).NE.2)GO TO 1360
      IF(NPEN(L,J).LE.2)GO TO 1330
      X=ARRAY(I,J)-HTEMP(2*J)
      IF(ABS(X).LT.0.001)GO TO 1310
      A=(HTEMP(2*J-1)-ARRAY(L,J))/X
      IF(A.LT.0.)GO TO 1310
      FRAC1=A/(1.+A)
      GO TO 1320
 1310 FRAC1=1.
 1320 NPEN(I,J)=1
 1330 IF(J.EQ.1)GO TO 1450
      IF(NPEN(I,J-1).LE.2)GO TO 1450
      IF(J.EQ.65)GO TO 1340
      HH1=(ARRAY(I,J)+ARRAY(I,J-1))/2.
      HH2=(ARRAY(L,J)+ARRAY(L,J-1))/2.
      IF(HH1.LT.ARRAY(L,J).AND.HH2.GT.ARRAY(I,J-1))GO TO 1340
      X=ARRAY(I,J)-HTEMP(2*J)
      IF(ABS(X).LT.0.001)GO TO 1340
      A=(HTEMP(2*J-2)-ARRAY(I,J-1))/X
      IF(A.LT.0.)GO TO 1340
      FRAC2=A/(1.+A)
      GO TO 1350
 1340 FRAC2=1.
 1350 NPEN(I,J)=NPEN(I,J)-2
      GO TO 1450
 1360 IF(NPEN(L,J).GT.2)GO TO 1390
      IF(I.EQ.64)GO TO 1390
      H=(ARRAY(L,J)+ARRAY(L,J+1))/2.
      IF(ARRAY(I,J).LE.H)GO TO 1390
      H=(HTEMP1(2*J+1)+HTEMP1(2*J-1))/2.
      X=HTEMP1(2*J+1)-ARRAY(I,J)
      IF(ABS(X).LT.0.001)GO TO 1370
      A=(ARRAY(L,J)-H)/X
```

```
      IF(A.LT.0.)GO TO 1370
      FRAC1=A/(1.+A)
      GO TO 1380
1370  FRAC1=1.
1380  NPEN(I,J)=4
1390  IF(J.EQ.1)GO TO 1450
      IF(NPEN(I,J-1).GT.2)1450
      X=HTEMP(2*J)-ARRAY(I,J)
      HH1=(ARRAY(I,J)+ARRAY(I,J-1))/2.
      IF(HH1.LT.HTEMP(2*J-1))X=2*HTEMP(2*J-1)-HTEMP(2*J-2)-ARRAY(I,J)
      IF(ABS(X).LT.0.001)GO TO 1400
      A=(ARRAY(I,J-1)-HTEMP(2*J-2))/X
      IF(A.LT.0.)GO TO 1400
      FRAC2=A/(1.+A)
      GO TO 1410
1400  FRAC2=1.
1410  NPEN(I,J)=NPEN(I,J)+2
1450  TEMP(I,J)=1000.*FLOAT(IFIX(100.*FRAC2))
      TEMP(I,J)=TEMP(I,J)+FLOAT(IFIX(100.*FRAC1))
1500  CONTINUE
      DO 1510 J=1,129
      HTEMP1(J)=HTEMP(J)
1510  CONTINUE
140   CONTINUE
      CALL PLOTS(IBUF,5000,7)
      CALL FACTOR(0.7)
      CALL SYMBOL(1.,8.,.14,CHAR,90.,48)
      CALL PLOT(5.,8.,-3)
      DO 180 I=65,1,-1
      DINCX=FLOAT(65-I)*GRIDX/2.
      N=1
      ROW(1,N)=DINCX
      ROW(2,N)=ARRAY(I,1)
      IPEN(N)=3
      N=N+1
      DO 170 J=2,65
      DINCX=DINCX+GRIDX
```

```
      IF(NPEN(I,J).NE.2)GO TO 1600
      ROW(1,N)=DINCX
      ROW(2,N)=ARRAY(I,J)
      IPEN(N)=2
      N=N+1
      GO TO 170
1600  IF(NPEN(I,J).NE.1)GO TO 1610
      ROW(1,N)=DINCX
      ROW(2,N)= ARRAY(I,J)
      IPEN(N)=2
      N=N+1
      GO TO 170
1610  IF(NPEN(I,J).GT.2)GO TO 1620
      FRAC2=TEMP(I,J)/100000.
      X=DINCX-(1.-FRAC2)*GRIDX
      Y=ARRAY(I,J-1)+(ARRAY(I,J)-ARRAY(I,J-1))*FRAC2
      ROW(1,N)=X
      ROW(2,N)=Y
      IPEN(N)=3
      N=N+1
      ROW(1,N)=DINCX
      ROW(2,N)=ARRAY(I,J)
      IPEN(N)=2
      N=N+1
      GO TO 170
1620  IF(NPEN(I,J).LT.5)GO TO 170
      FRAC2=TEMP(I,J)/100000.
      X=DINCX-(1.-FRAC2)*GRIDX
      Y=ARRAY(I,J-1)+(ARRAY(I,J)-ARRAY(I,J-1))*FRAC2
      ROW(1,N)=X
      ROW(2,N)=Y
      IPEN(N)=2
      N=N+1
170   CONTINUE
      N=N-1
      IF(MOD(I,2).EQ.0)GO TO 175
      DO 171 K=1,N
171   CALL PLOT(ROW(1,K),ROW(2,K),IPEN(K))
      GO TO 180
```

```
175   NN=N-1
      DO 176 K=1,NN
176   IF(IPEN(K).NE.IPEN(K+1))IPEN(K)=IPEN(K+1)
      IPEN(N)=3
      DO 177 K=N,1,-1
177   CALL PLOT(ROW(1,K),ROW(2,K),IPEN(K))
180   CONTINUE
      XP=GRIDX*64.
      XQ=96.*GRIDX
      YP=64.*GRIDY
      CALL PLOT(XQ,YP,2)
      CALL PLOT(XP,0.,2)
      CALL PLOT(XP,ARRAY(65,65),2)
      CALL PLOT(XP,0.,3)
      CALL PLOT(0.,0.,2)
      CALL PLOT(0.,ARRAY(65,1),3)
      IF(NLINE.EQ.1)GO TO 250
      DO 230 J=1,65
      DELTAX=FLOAT(J-1)*GRIDX
      N=1
      ROW(1,N)=DELTAX
      ROW(2,N)=ARRAY(65,J)
      IPEN(N)=3
      N=N+1
      DO 200 I=64,1,-1
      DELTAX=DELTAX+GRIDX/2.
      IF(NPEN(I,J).NE.2)GO TO 1700
      ROW(1,N)=DELTAX
      ROW(2,N)=ARRAY(I,J)
      IPEN(N)=2
      N=N+1
      GO TO 200
1700  IF(NPEN(I,J).NE.0)GO TO 1710
      ROW(1,N)=DELTAX
      ROW(2,N)=ARRAY(I,J)
      IPEN(N)=2
      N=N+1
      GO TO 200
```

```
1710  IF(NPEN(I,J).GT.2)GO TO 1720
      FRAC1=AMOD(TEMP(I,J),1000.)/100.
      X=DELTAX-(1.-FRAC1)*GRIDX/2.
      Y=ARRAY(I+1,J)+(ARRAY(I,J)-ARRAY(I+1,J))*FRAC1
      ROW(1,N)=X
      ROW(2,N)=Y
      IPEN(N)=3
      N=N+1
      ROW(1,N)=DELTAX
      ROW(2,N)=ARRAY(I,J)
      IPEN(N)=2
      N=N+1
      GO TO 200
1720  IF(NPEN(I,J).EQ.3.OR.NPEN(I,J).EQ.5)GO TO 200
      FRAC1=AMOD(TEMP(I,J),1000.)/100.
      X=DELTAX-(1.-FRAC1)*GRIDX/2.
      Y=ARRAY(I+1,J)+(ARRAY(I,J)-ARRAY(I+1,J))*FRAC1
      ROW(1,N)=X
      ROW(2,N)=Y
      IPEN(N)=2
      N=N+1
200   CONTINUE
      N=N-1
      IF(MOD(J,2).EQ.0)GO TO 210
      DO 205 K=1,N
205   CALL PLOT(ROW(1,K),ROW(2,K),IPEN(K))
      GO TO 230
210   NN=N-1
      DO 215 K=1,NN
215   IF(IPEN(K).NE.IPEN(K+1))IPEN(K)=IPEN(K+1)
      IPEN(N)=3
      DO 220 K=N,1,-1
220   CALL PLOT(ROW(1,K),ROW(2,K),IPEN(K))
230   CONTINUE
250   CALL PLOT(30.,-20.,999)
      STOP
      END
```